# CS4410/11: Operating Systems

## CPU Scheduling

Rachit Agarwal
Anne Bracy

# CPU Scheduling — Problem Description

**You have multiple CPU tasks (processes) to execute!**

- **Which one to execute next?**

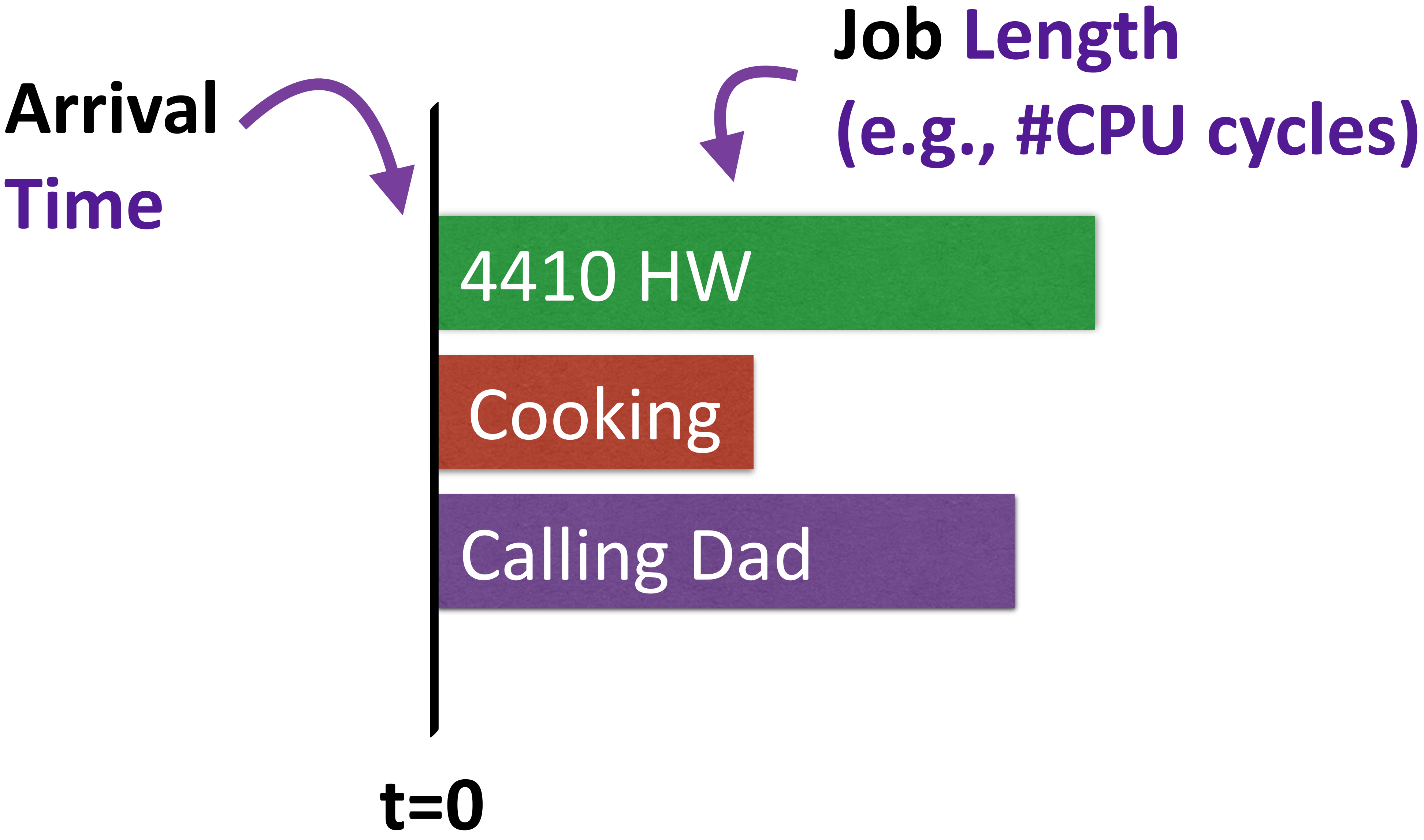
4410 HW

Cooking

Calling Dad

- Answer obvious in this example
- But, usually not!

# CPU Scheduling — Example 1 (Idealized)



Arrival Time

Job Length (e.g., #CPU cycles)

4410 HW

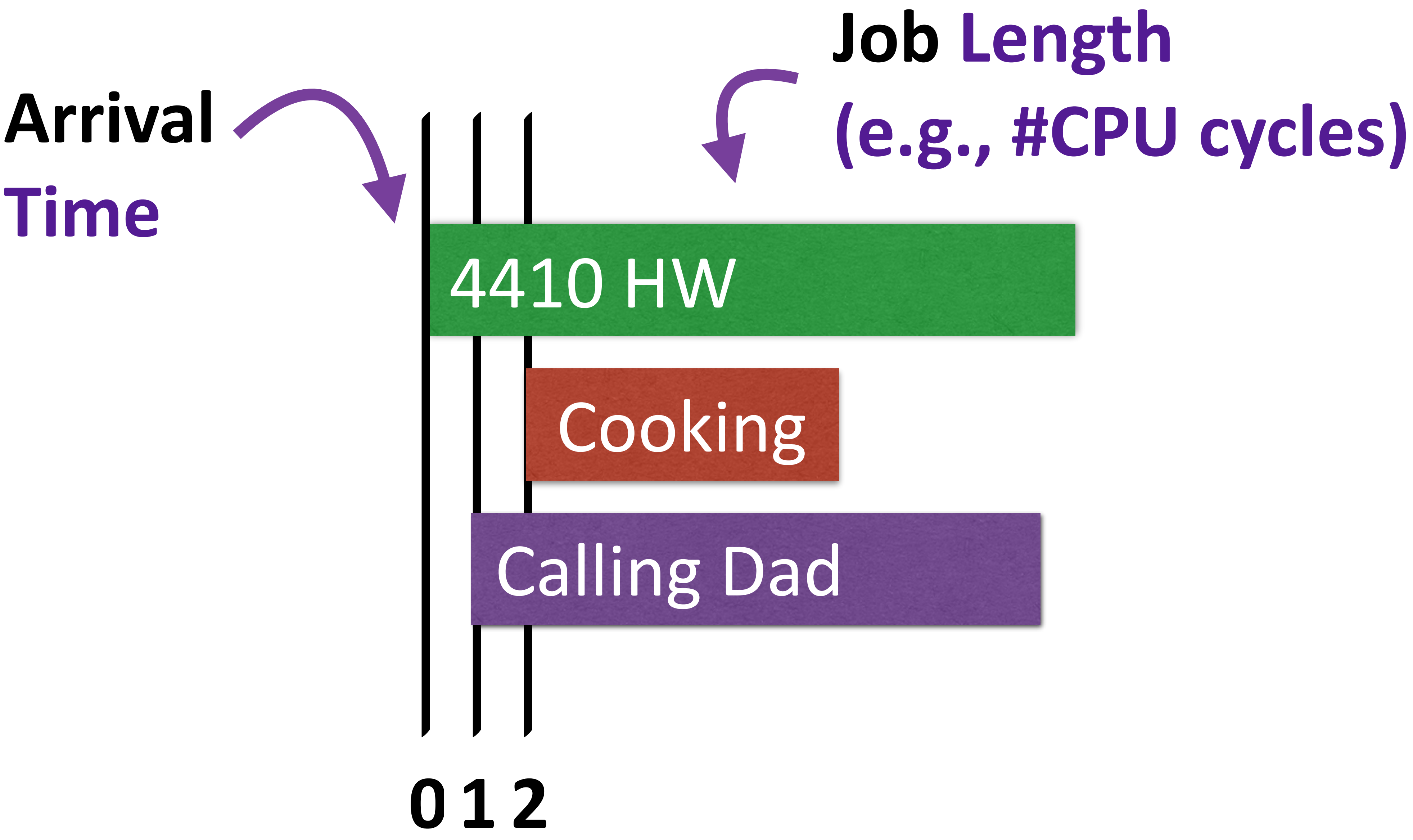
Cooking

Calling Dad

t=0

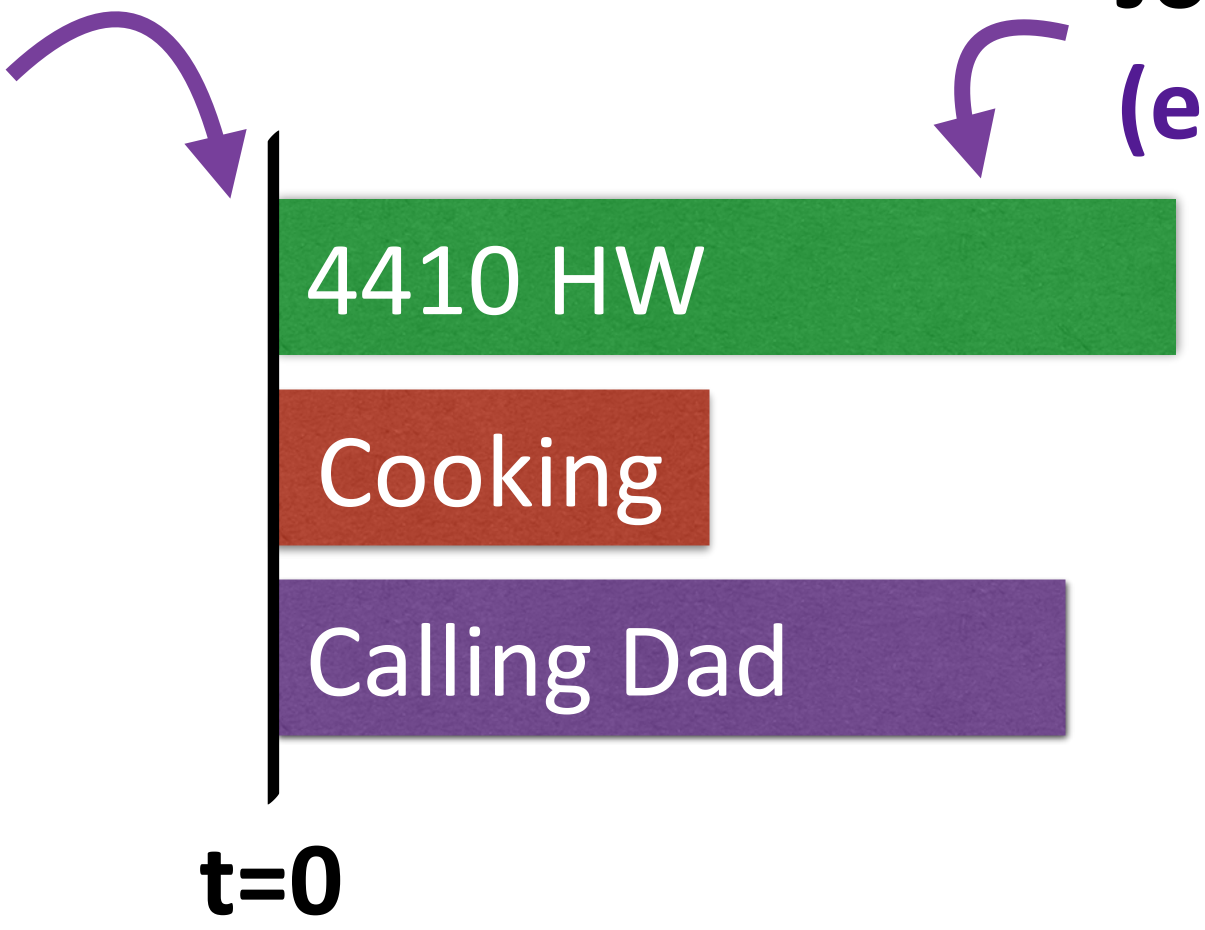
**Each job may have a different length**

# CPU Scheduling — Example 1 (Idealized)



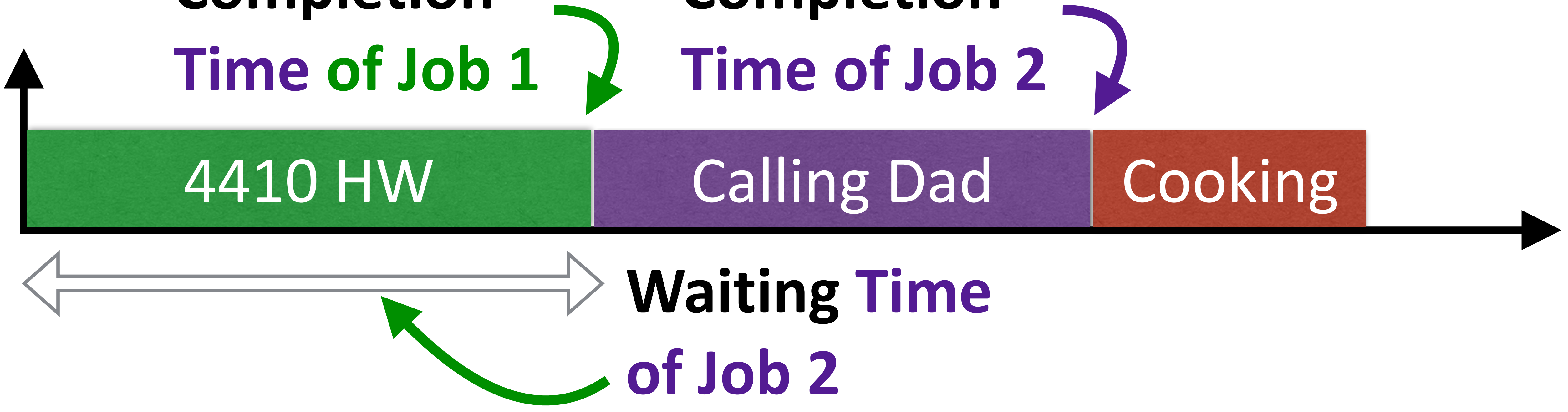
**Each job may arrive at different time**

# CPU Scheduling — Example 1 (Idealized)



Arrival Time

Job Length (e.g., #CPU cycles)



Completion Time of Job 1

Completion Time of Job 2

Waiting Time of Job 2

# CPU Scheduling — Why is it important?

**Problem encountered in many setting. Similar principles!**

- **Street**
    - Which car should move next?
- **Supermarkets**
    - Which customer to help next?
- **Airports**
    - Which plane should land (or fly) next?
- **Hospitals**
    - Which patient to attend next?

# CPU Scheduling — Why is it important? [Cont.]

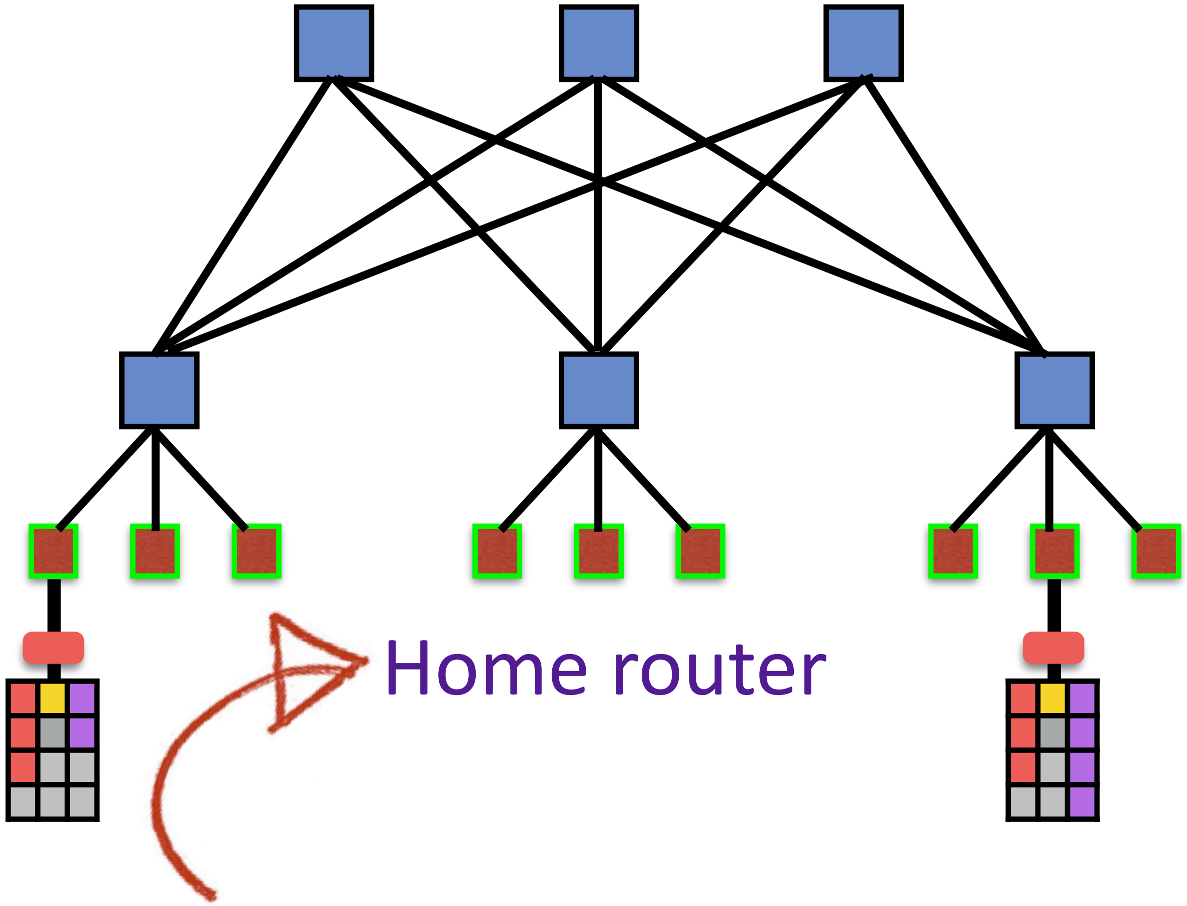
**Problem encountered in many setting. Similar principles!**

- **Within an OS**
    - CPU scheduler
    - Disk or I/O Scheduler
    - Network Scheduler
    - ….

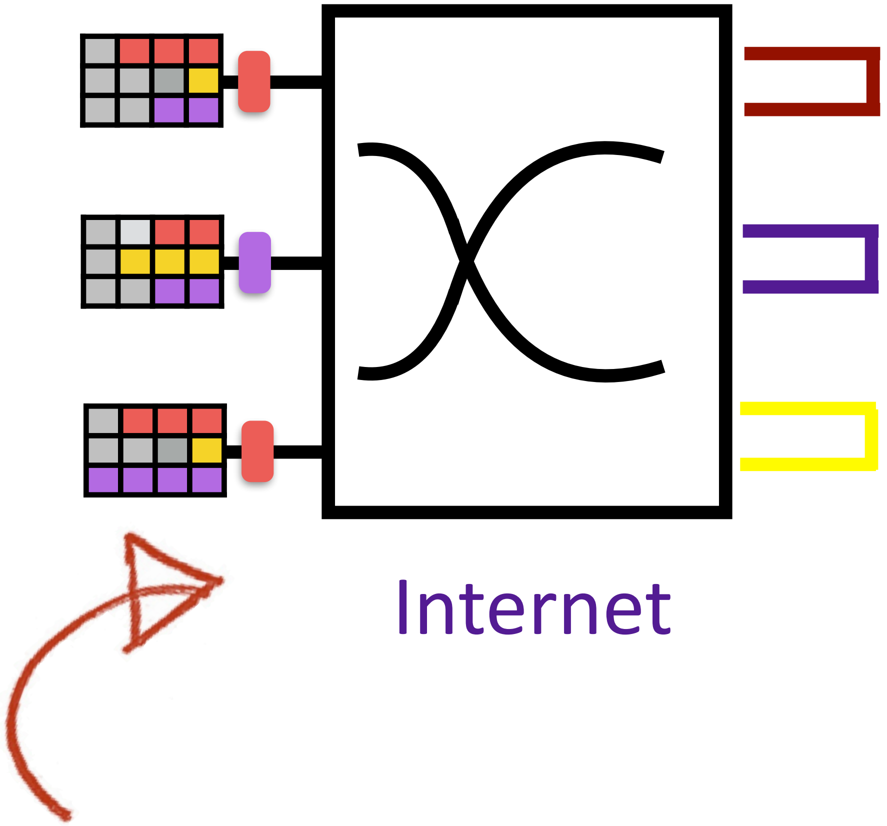
# CPU Scheduling — Why is it important? [Cont.]

**Problem encountered in many setting. Similar principles!**

- **Networks (Internet, Google, Facebook, ….)**



Which user to schedule next? (you or your roommate)



Which packet to schedule next? (Movies or Skype)

# CPU Scheduling — Why is it important? [Cont.]

**Problem encountered in many setting. Similar principles!**

- **Cloud (Amazon, Google, Facebook, ….)**
  - Thousands of machines
  - Different kind of customers can rent machines
  - Also, internal employees want to use machines
  - Who should use which machine and when?

**A very active area of research!!**

# CPU Scheduling — Lecture Plan

- Learn **about** various scheduling policies
- Learn how to **compare** different policies
- Learn how to **choose** between different policies
- **Strategy**:
  - A lot of examples
  - Active problem solving

# Operating Systems Design Principles

## Six principles we discussed in first lecture —

- Reliability
- Availability
- Security
- Privacy
- Portability
- Fairness
    - Taking it to an extreme: Starvation
    - **Whats even more extreme?**

# Operating Systems Design Principles

**Today, we will focus on two —**

- Reliability
- Availability
- Security
- Privacy
- Portability
- **Fairness**
    - Taking it to an extreme: **Starvation**

# Operating Systems Performance

- **Latency**
    - How long does a task take to complete?
- **Throughput**
    - #Tasks per unit time
- **Utilization**
    - Fraction of resources used over time
- **Scalability**
    - How does the performance change with size?
- **Predictability**
    - Consistency (over time) for an objective

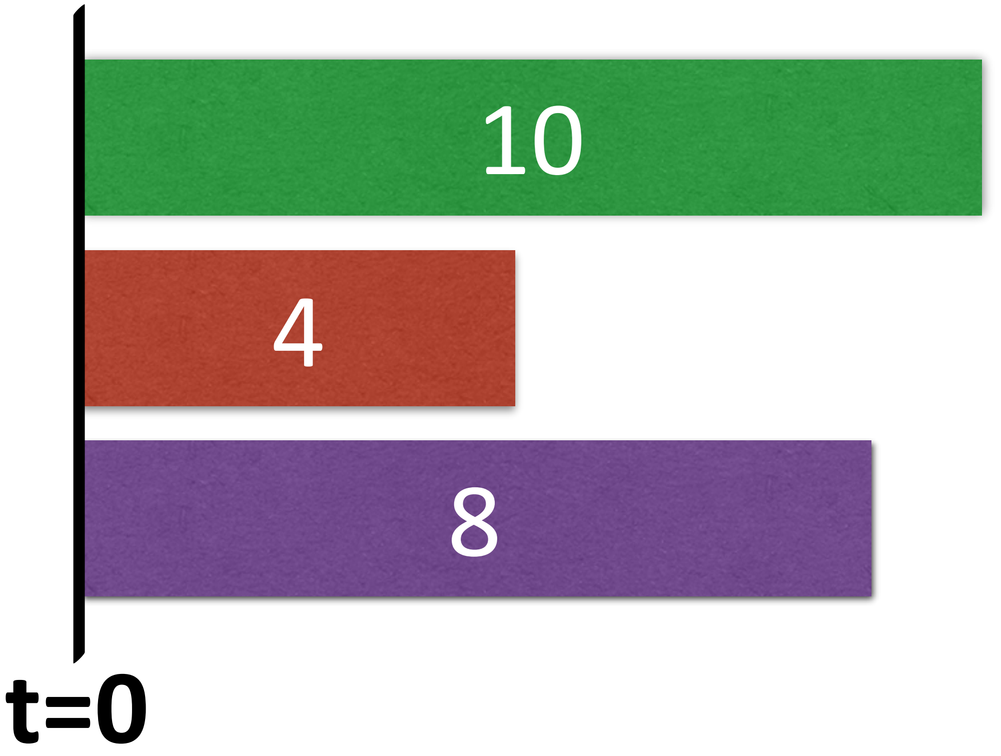
# Operating Systems Performance

- **Deadlines**
  - How many of the tasks meet their deadlines?

# CPU Scheduling — Latency

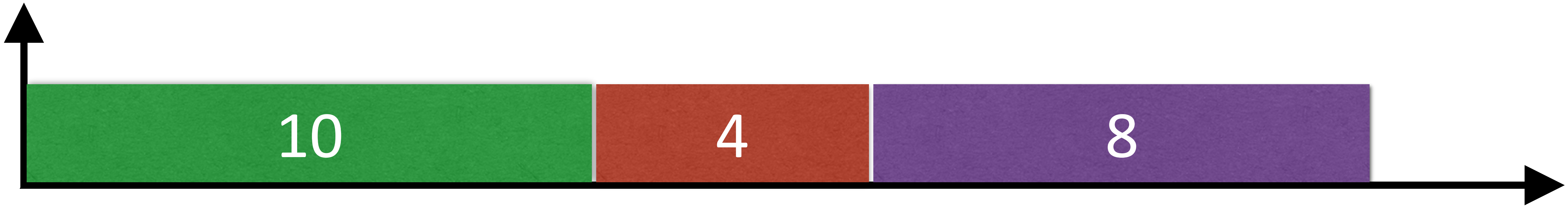
- **“Tail” Completion Time**
  - When does the last task complete?
- **Average Completion Time**
  - How long does it take to complete a task **on an average**?
- **High Percentile Completion Time**
  - How long does it take to complete 90% of the tasks?
- **Completion Time of “Small” Tasks**
- **Waiting time of Tasks**
- **….**

# Scheduling for “Tail” Latency — FIFO, FCFS

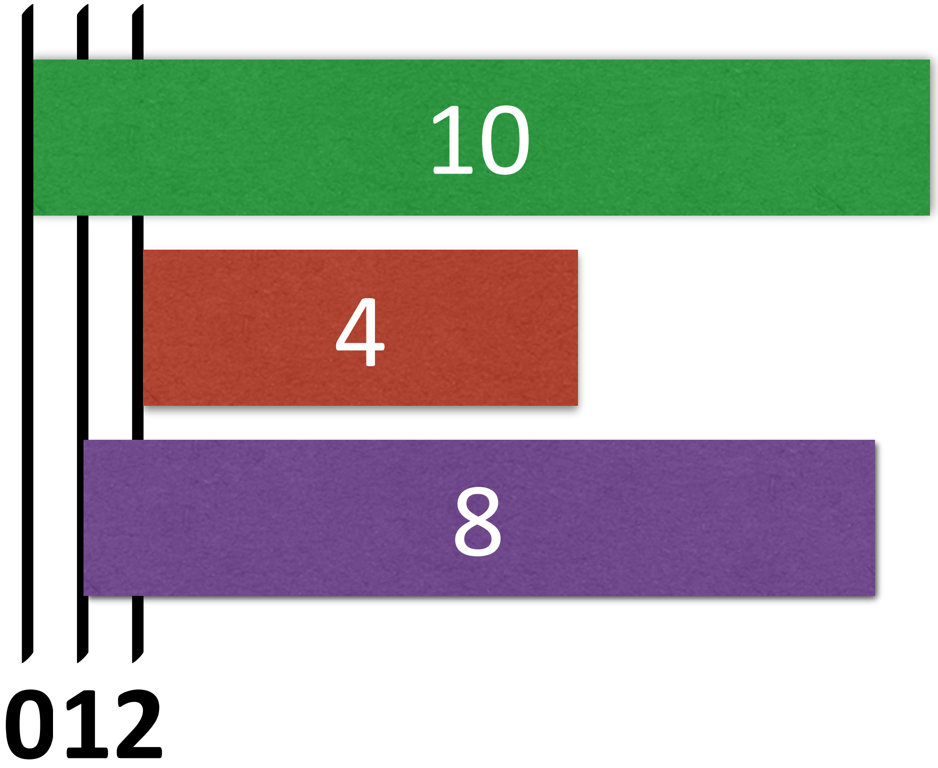


- **First-In First-Out**

- **Schedule?**

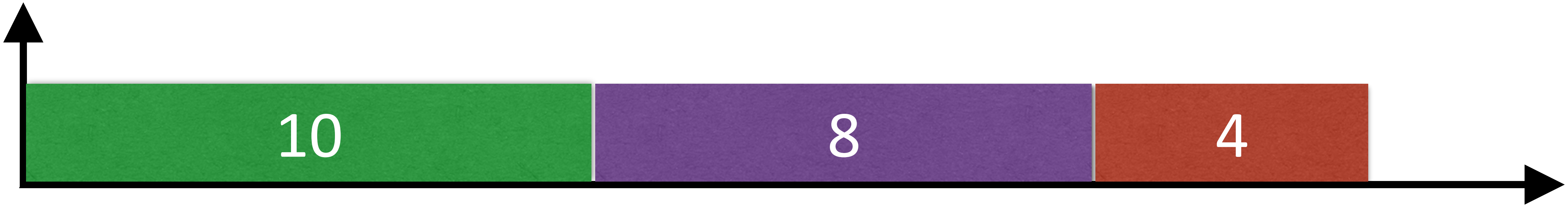


# Scheduling for "Tail" Latency — FIFO, FCFS



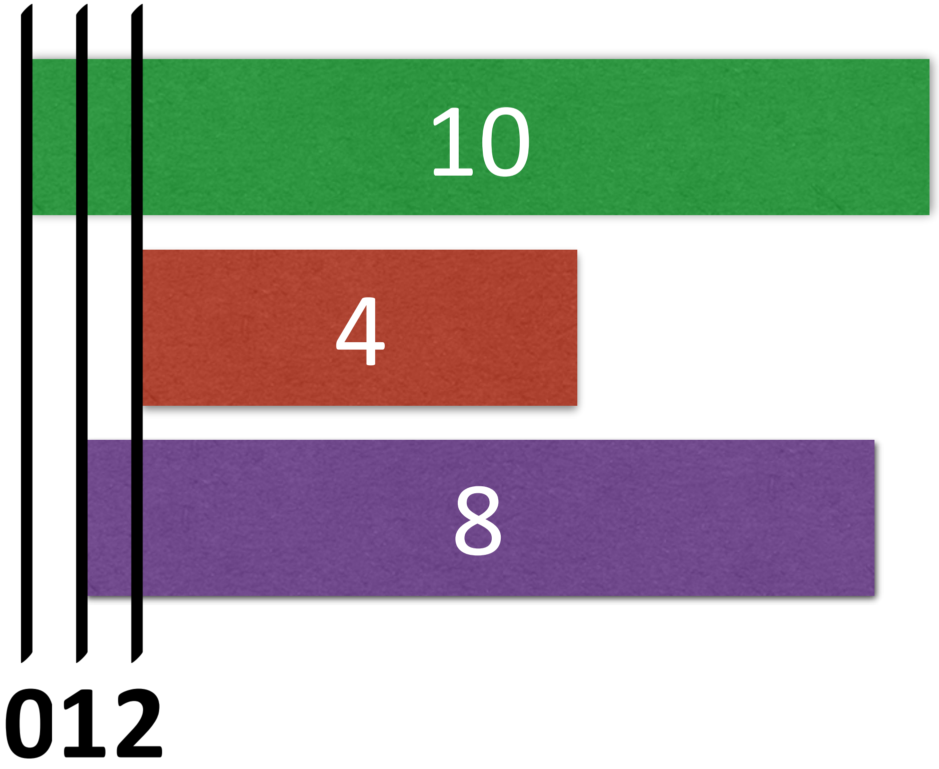
- **First-In First-Out**
- **When does it matter?**
- **Goods:** Simple
- **Not-so-goods: High ACT**

- **Schedule?**



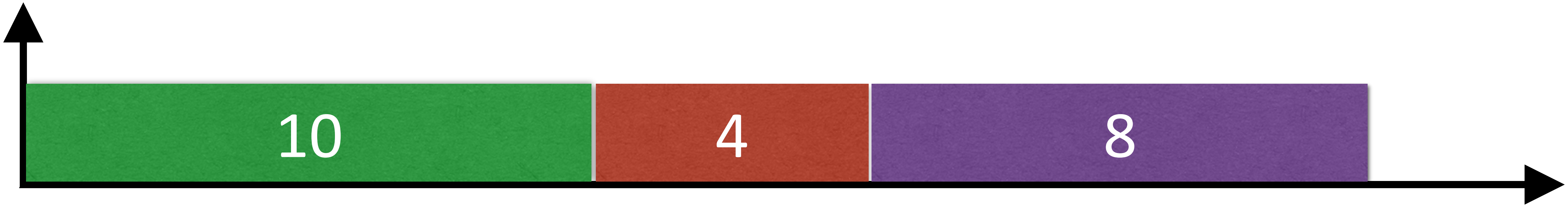
© Alamy

# Scheduling for "Tail" Latency — LIFO



- **Last-In First-Out**
- **Goods: ?**
- **Not-so-goods: High ACT**
- **Not-so-goods: Starvation**
- **Why?**

- **Schedule?**

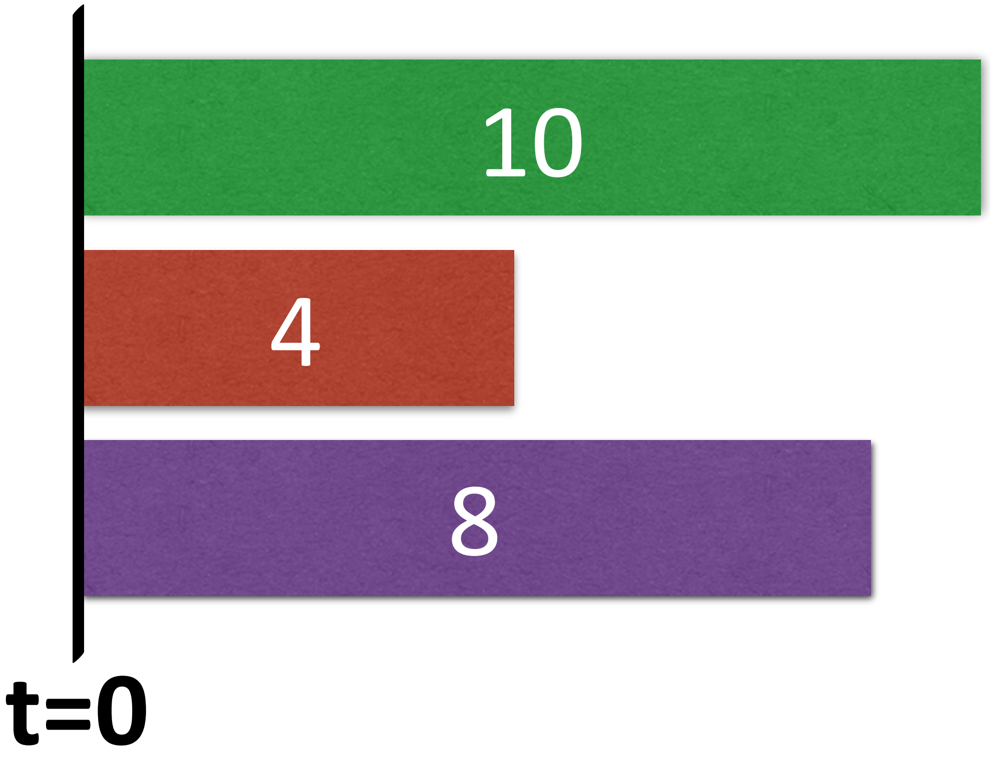


FRAGILE

# Undo™

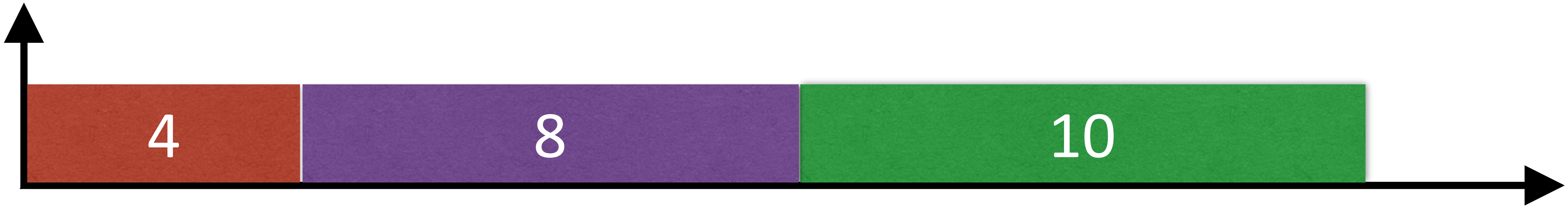
Bugs that took weeks now take minutes

# Scheduling for “Average” Latency — SJF

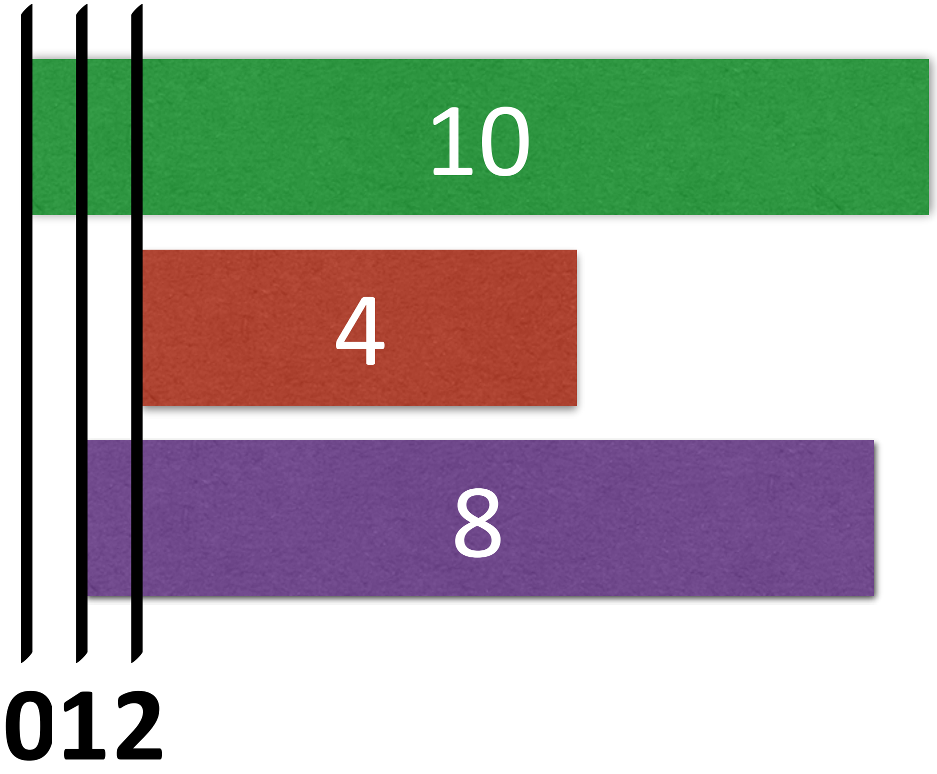


- **Shortest-Job First**
- **Goods:** Minimizes ACT
- **Not-so-goods: Starvation**
- **Why?**

- **Schedule?**

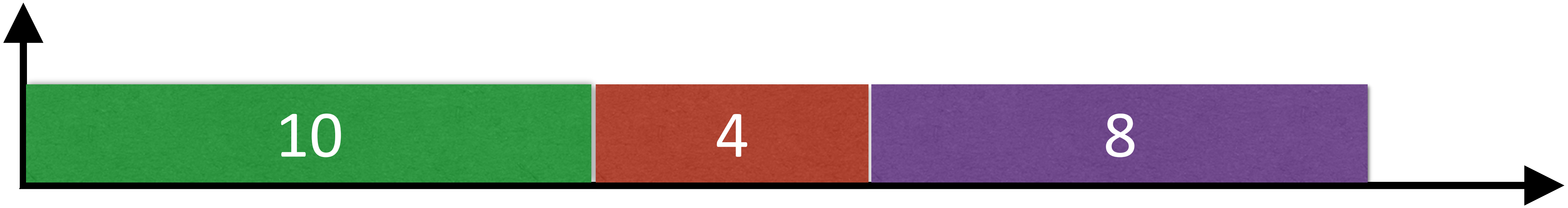


# Scheduling for “Average” Latency — SJF

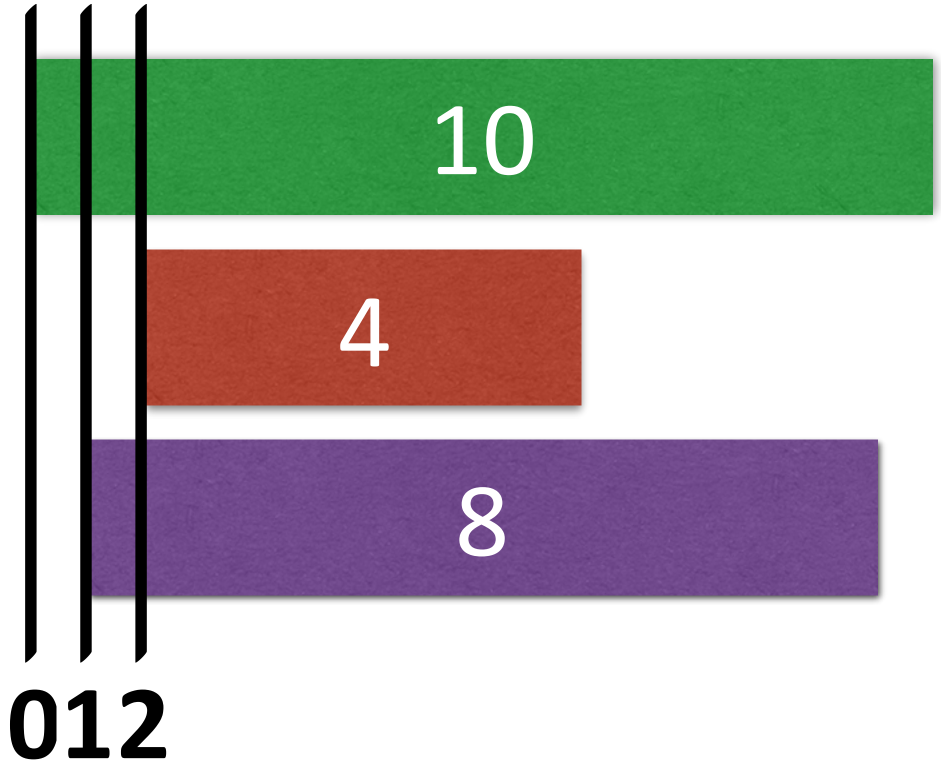


- **Shortest-Job First**
- **Goods:** Minimizes ACT
- **Not-so-goods: Starvation**
- **Optimal? Why, or why not?**

- **Schedule?**

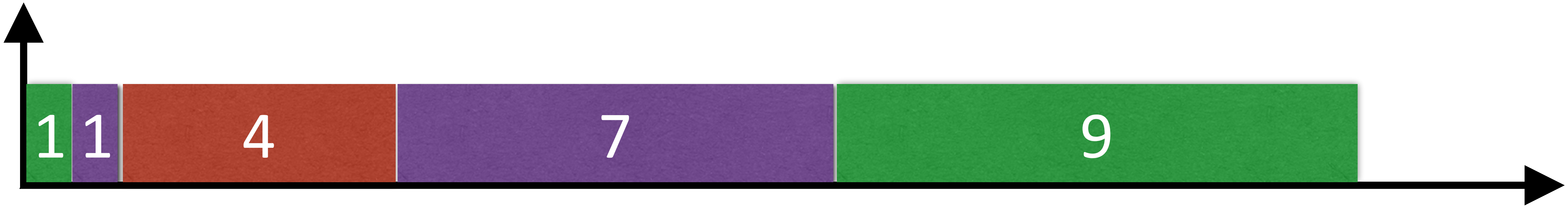


# Scheduling for “Average” Latency — (P)-SJF

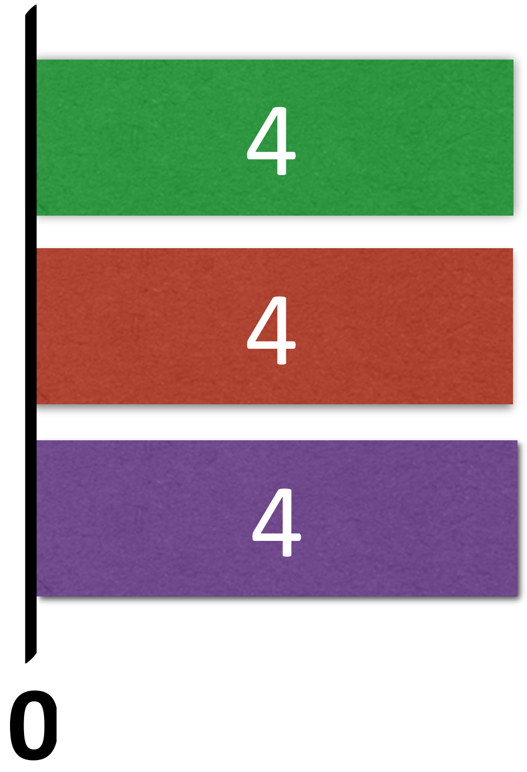


- Shortest Job First + **Preemption**
- **When is Preemption useful?**

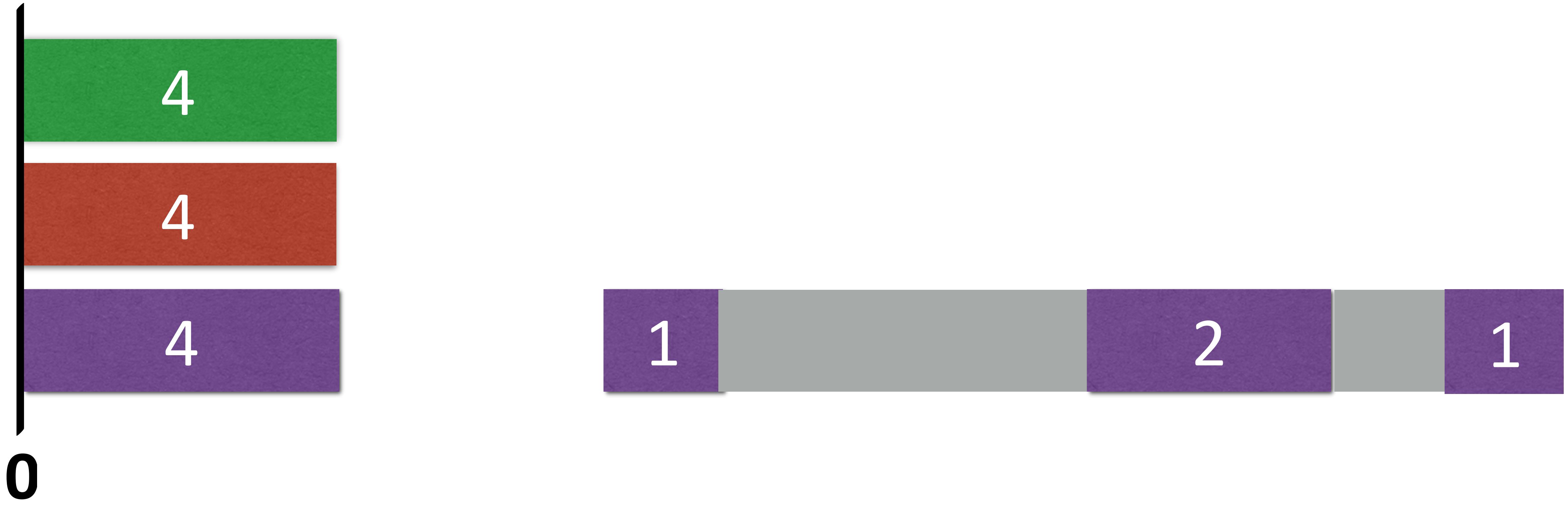
- **Schedule?**



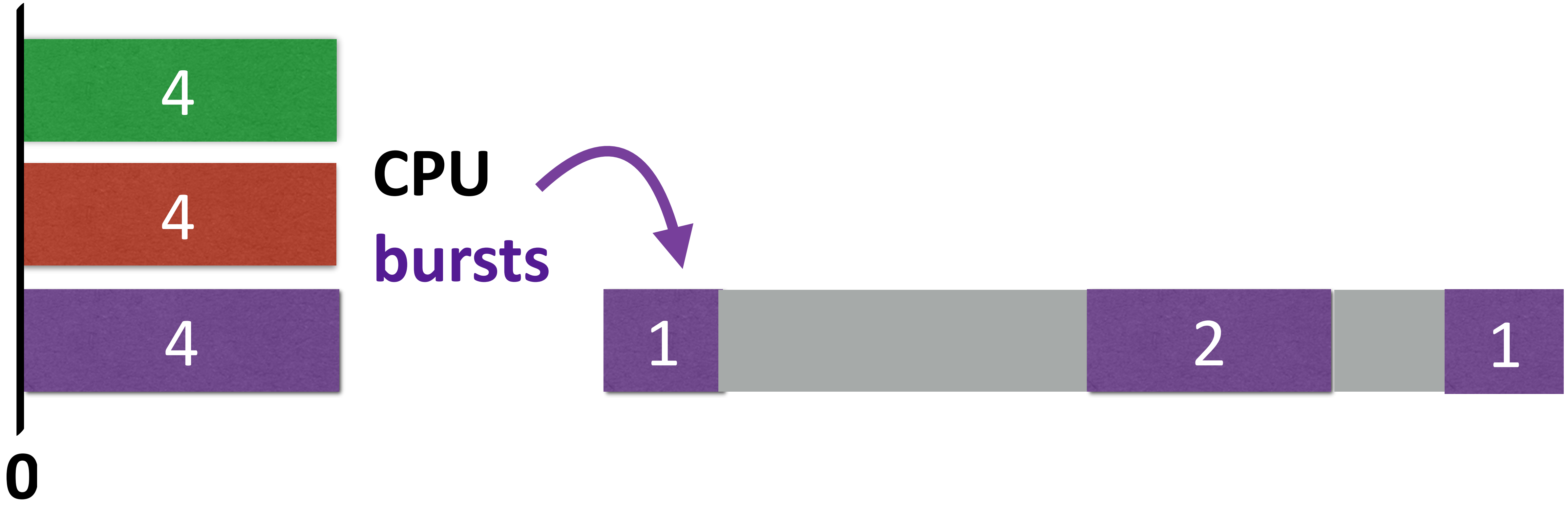
# Improved Resource Utilization — Preemption



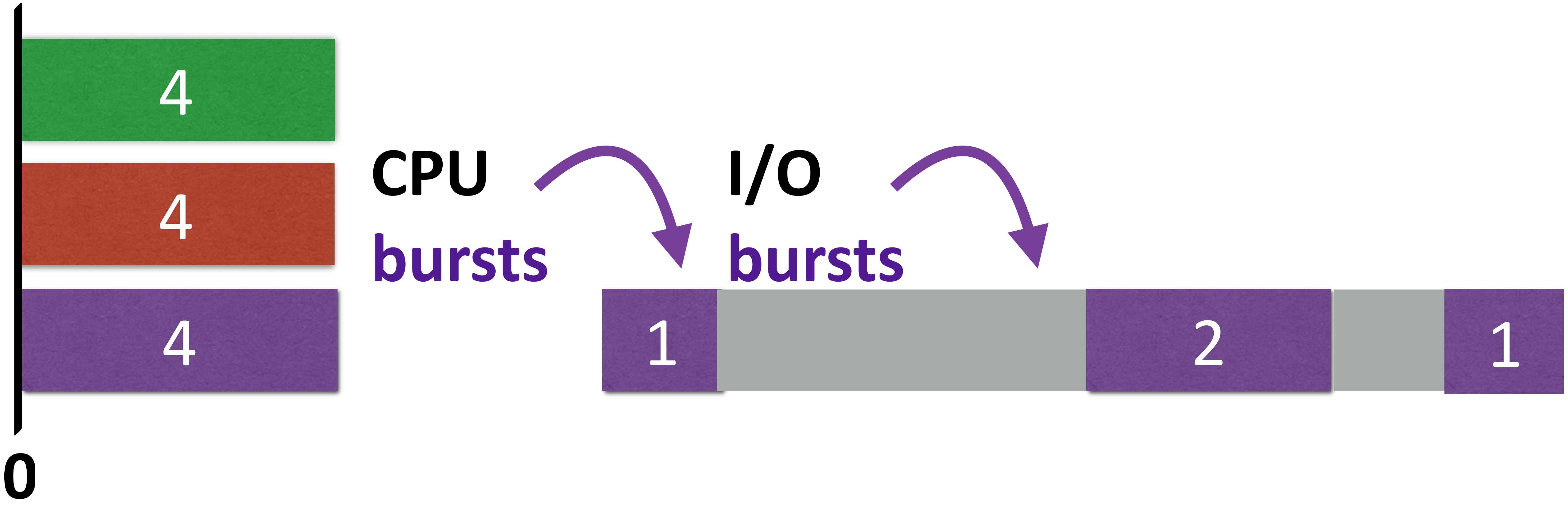
# Improved Resource Utilization — Preemption



# Improved Resource Utilization — Preemption



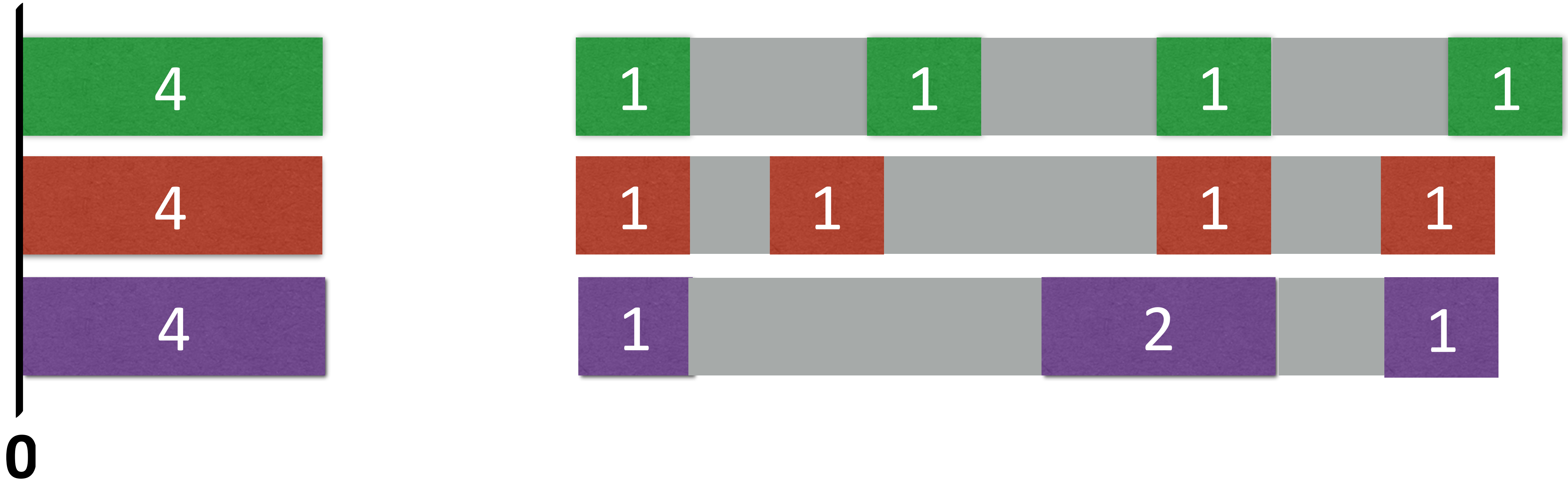
# Improved Resource Utilization — Preemption



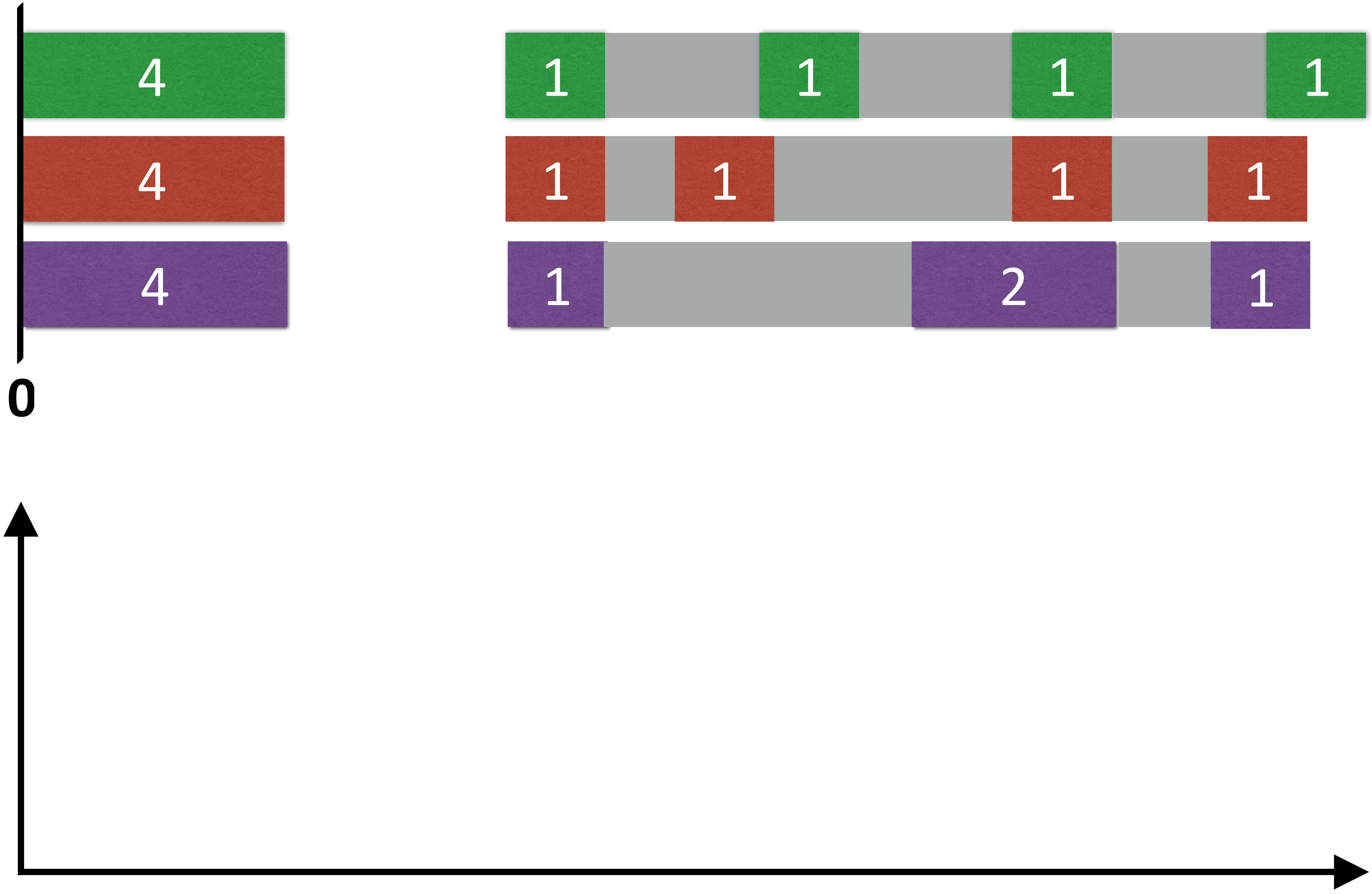
CPU bursts

I/O bursts

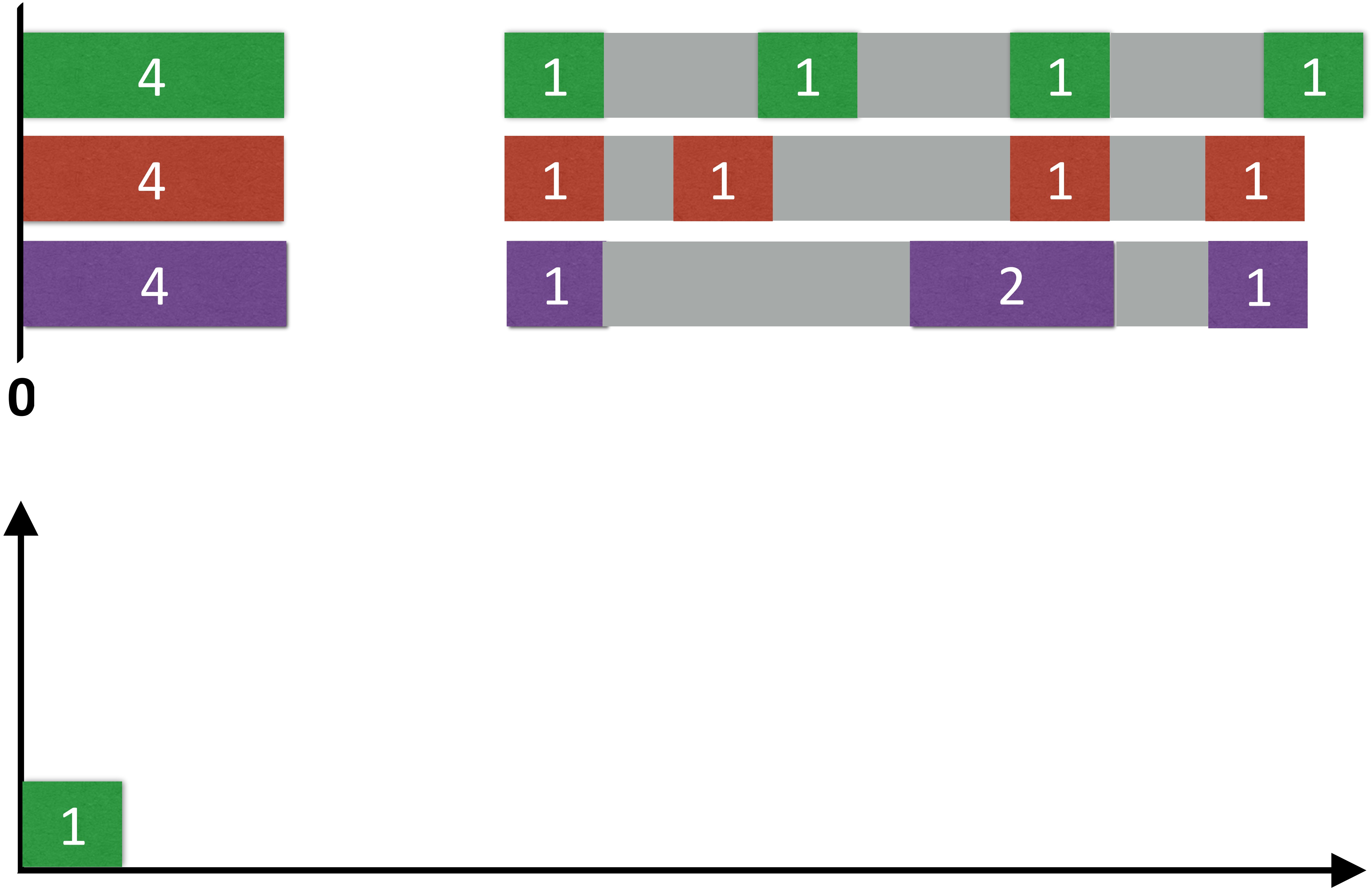
# Improved Resource Utilization — Preemption



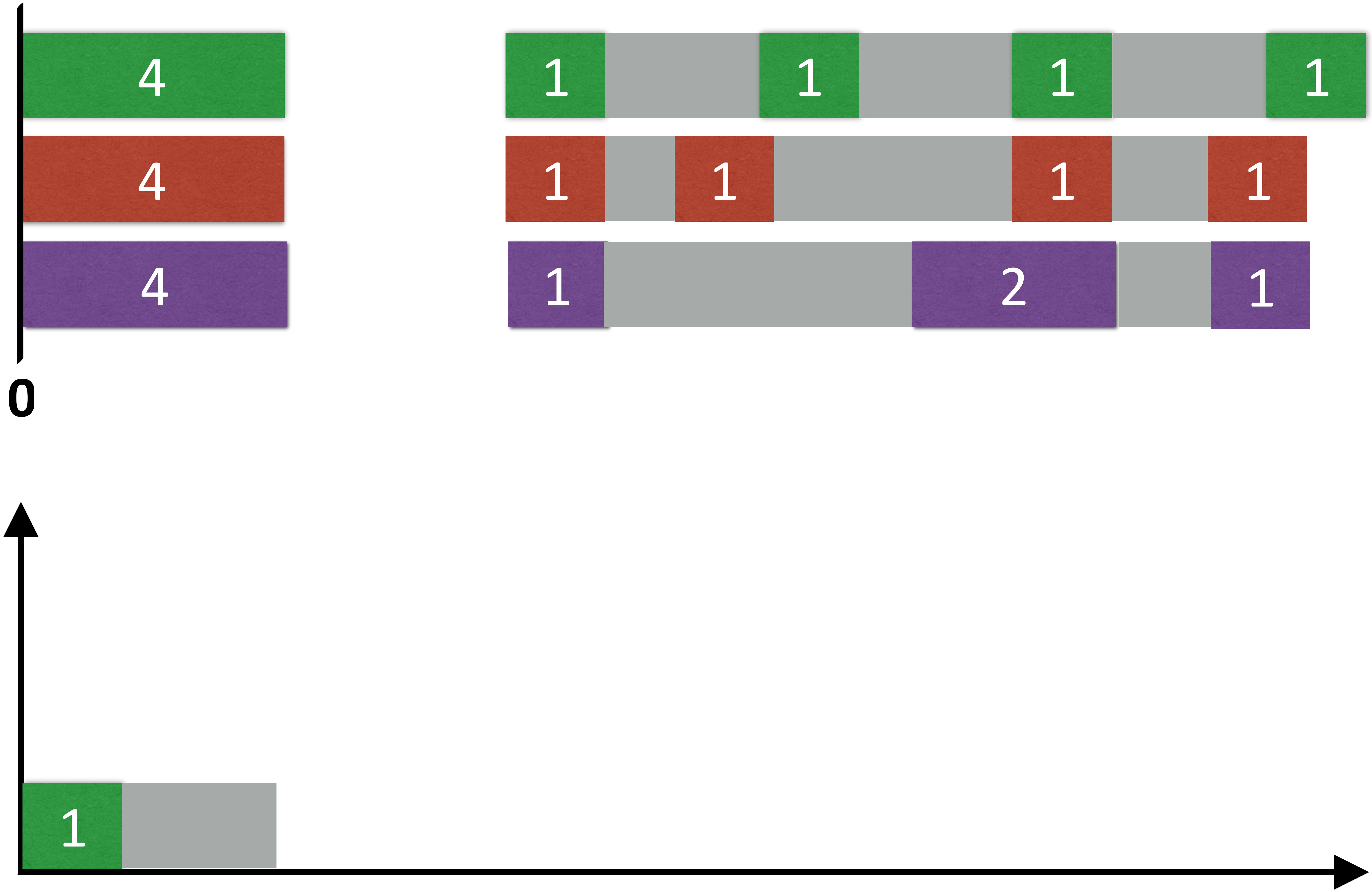
# Improved Resource Utilization — Preemption



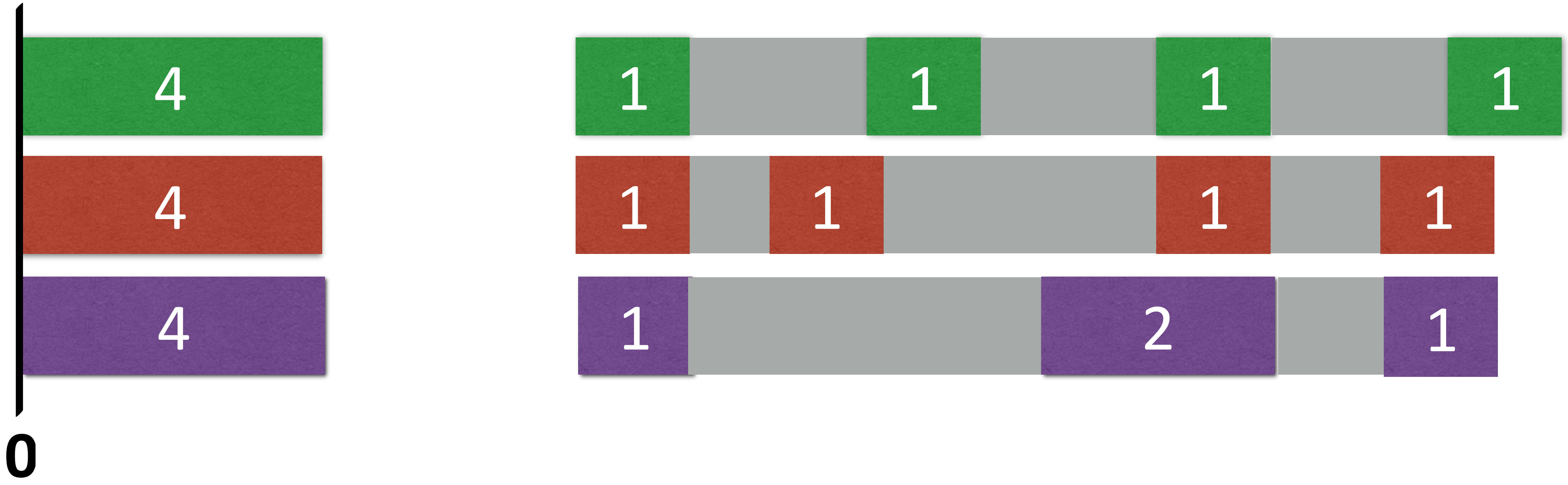
# Improved Resource Utilization — Preemption

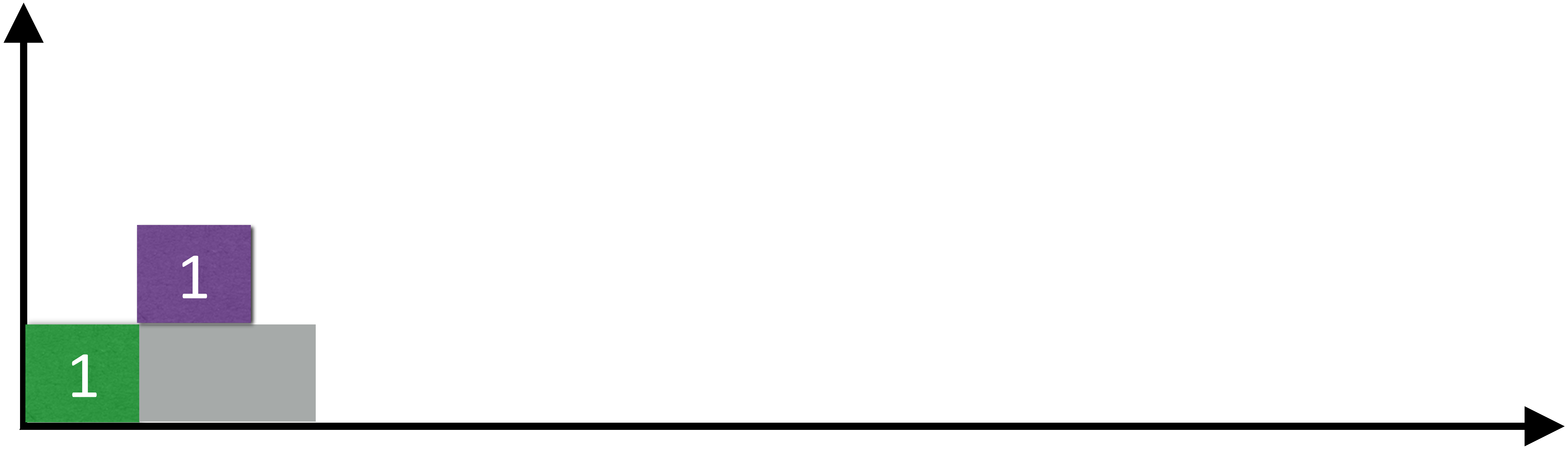


# Improved Resource Utilization — Preemption

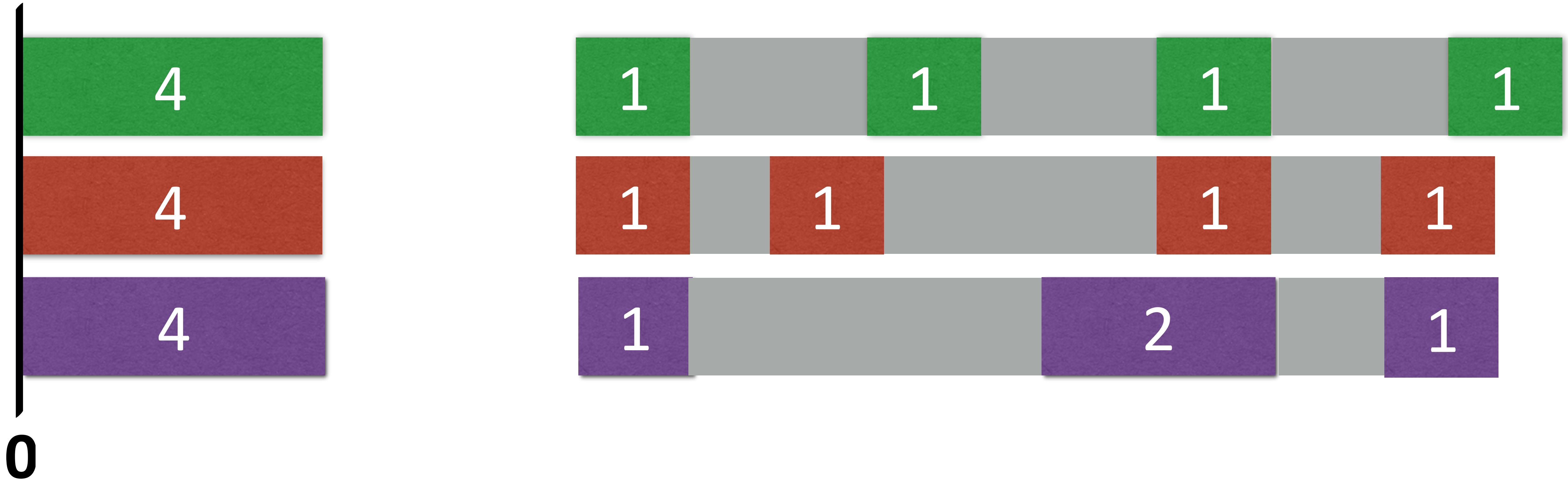


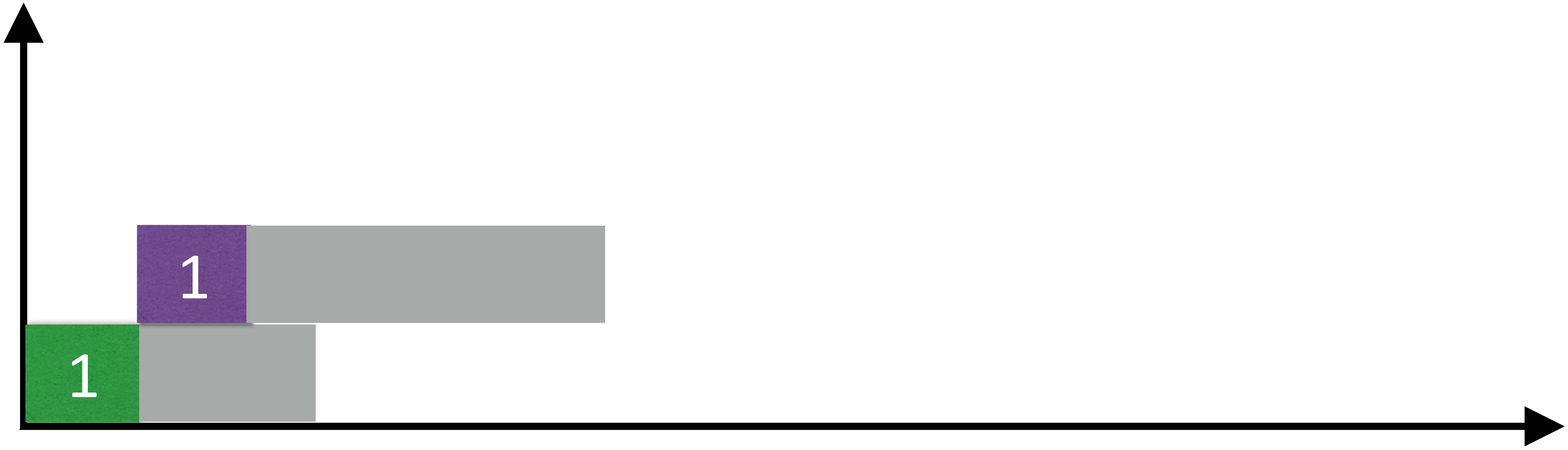
# Improved Resource Utilization — Preemption



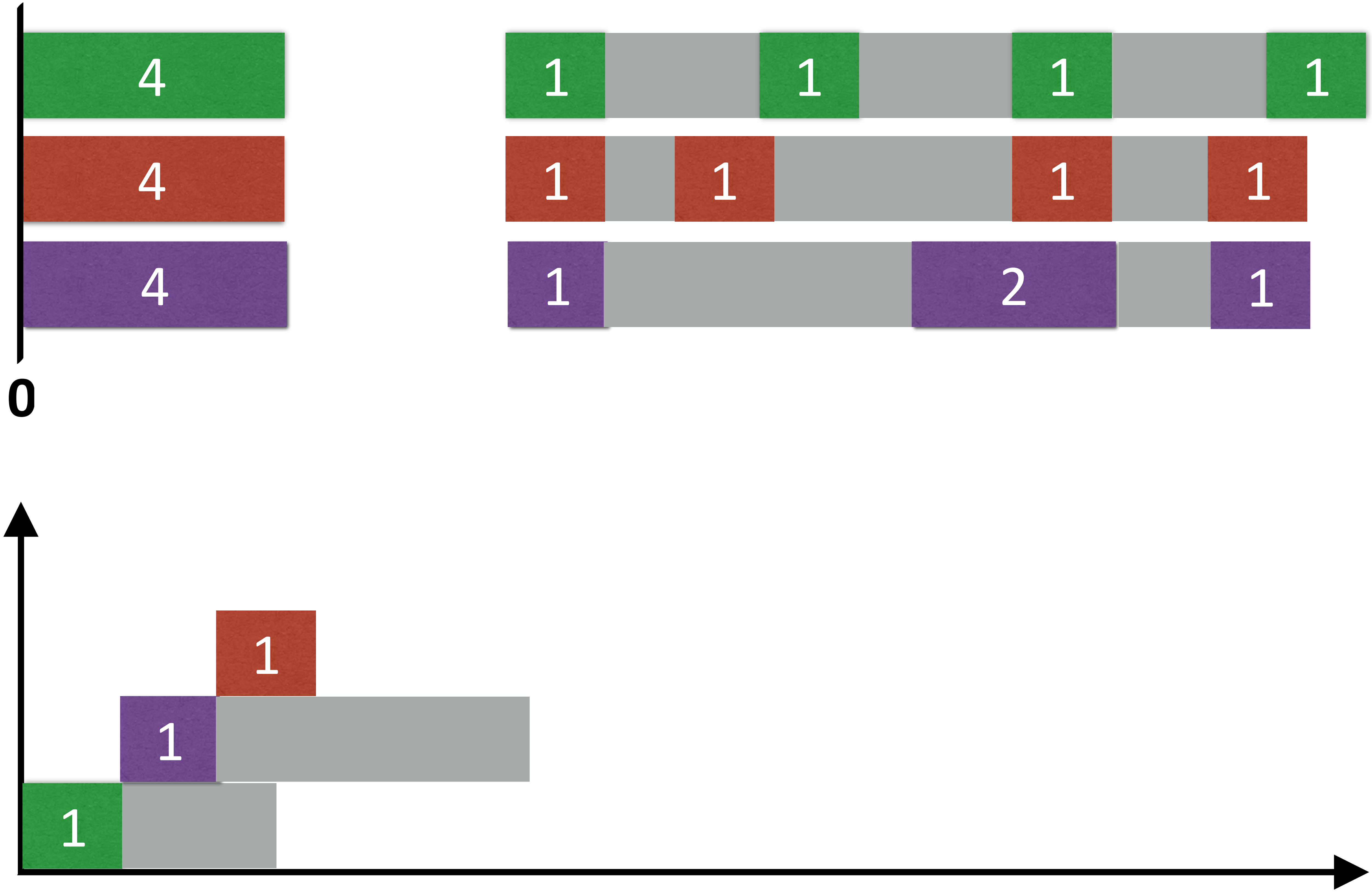


# Improved Resource Utilization — Preemption

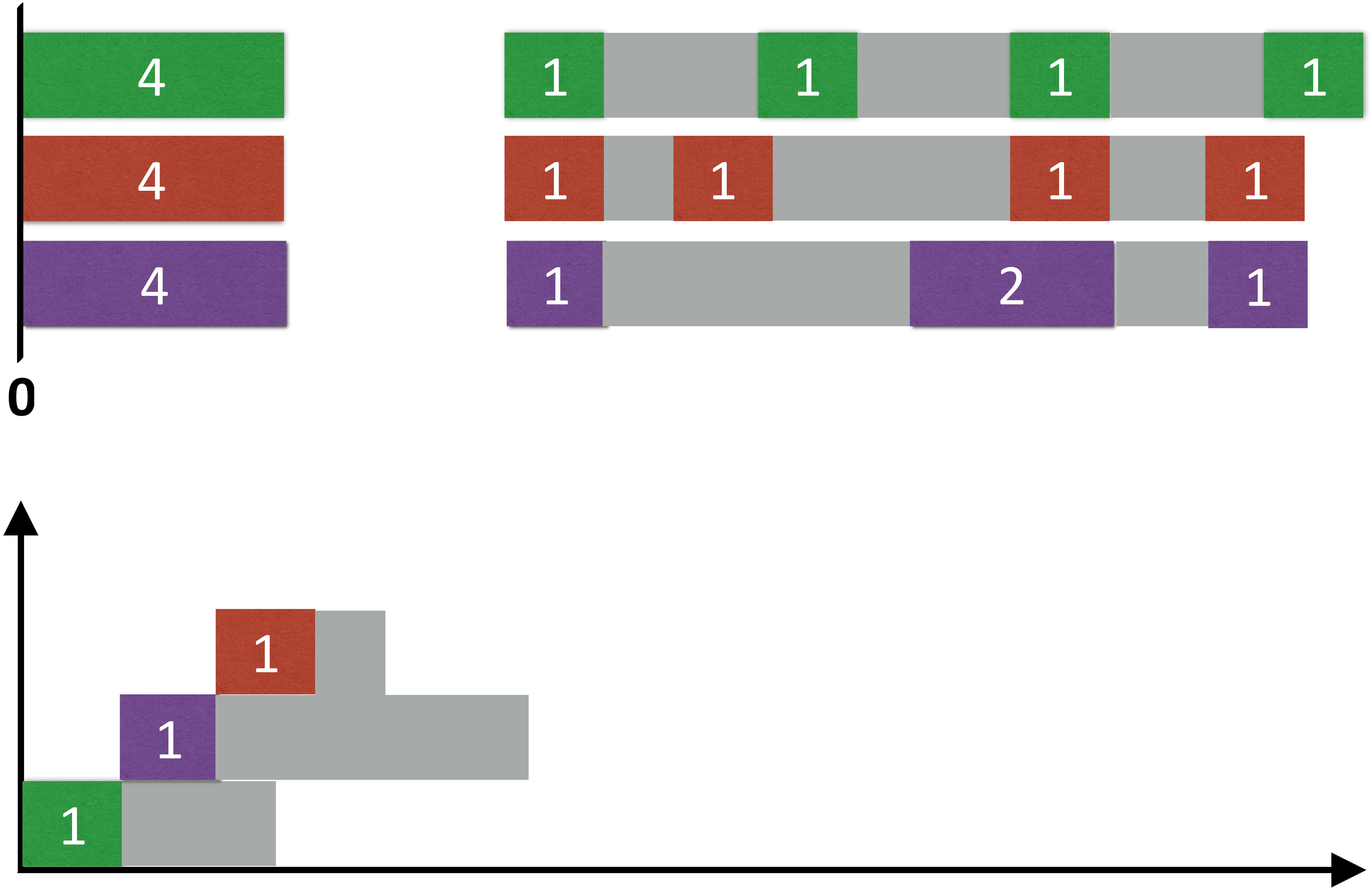




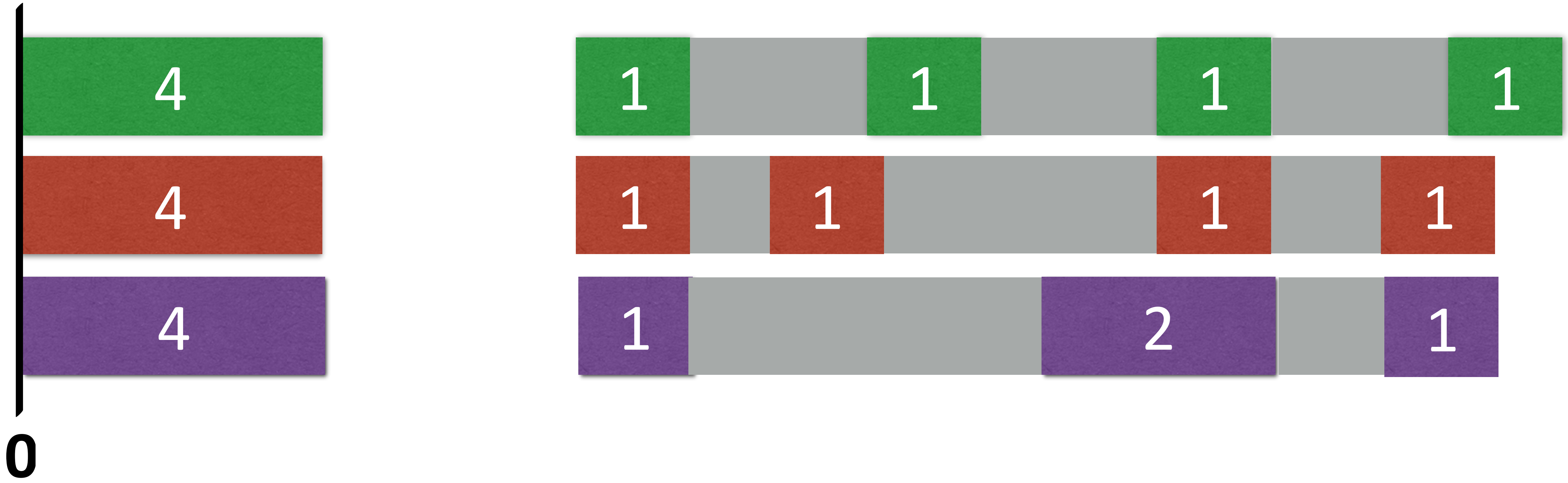
# Improved Resource Utilization — Preemption

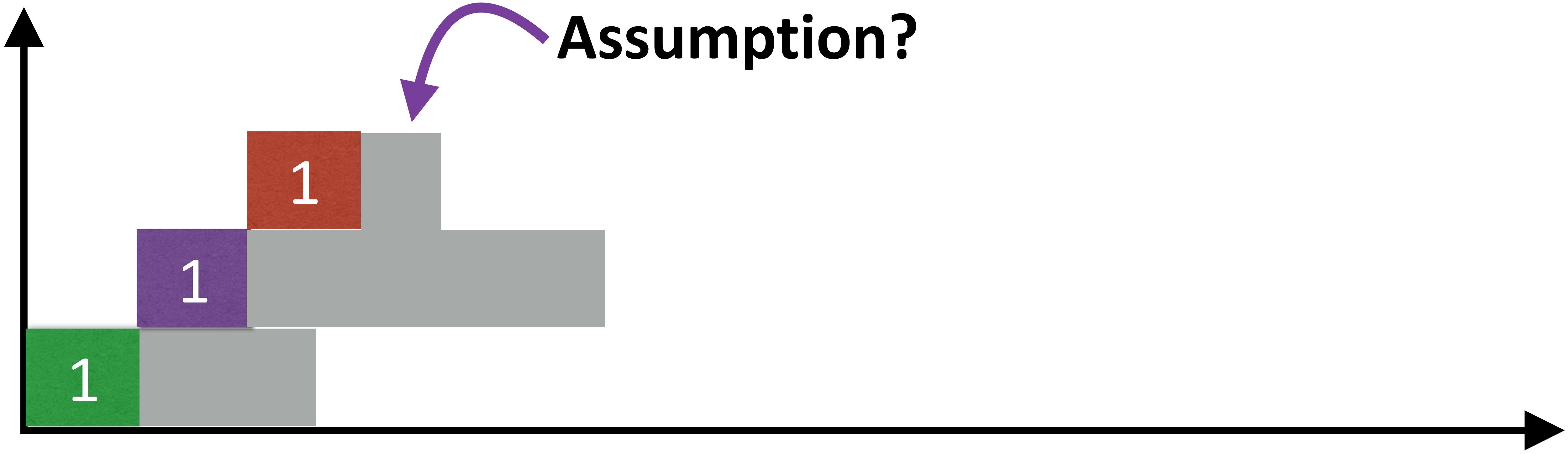


# Improved Resource Utilization — Preemption

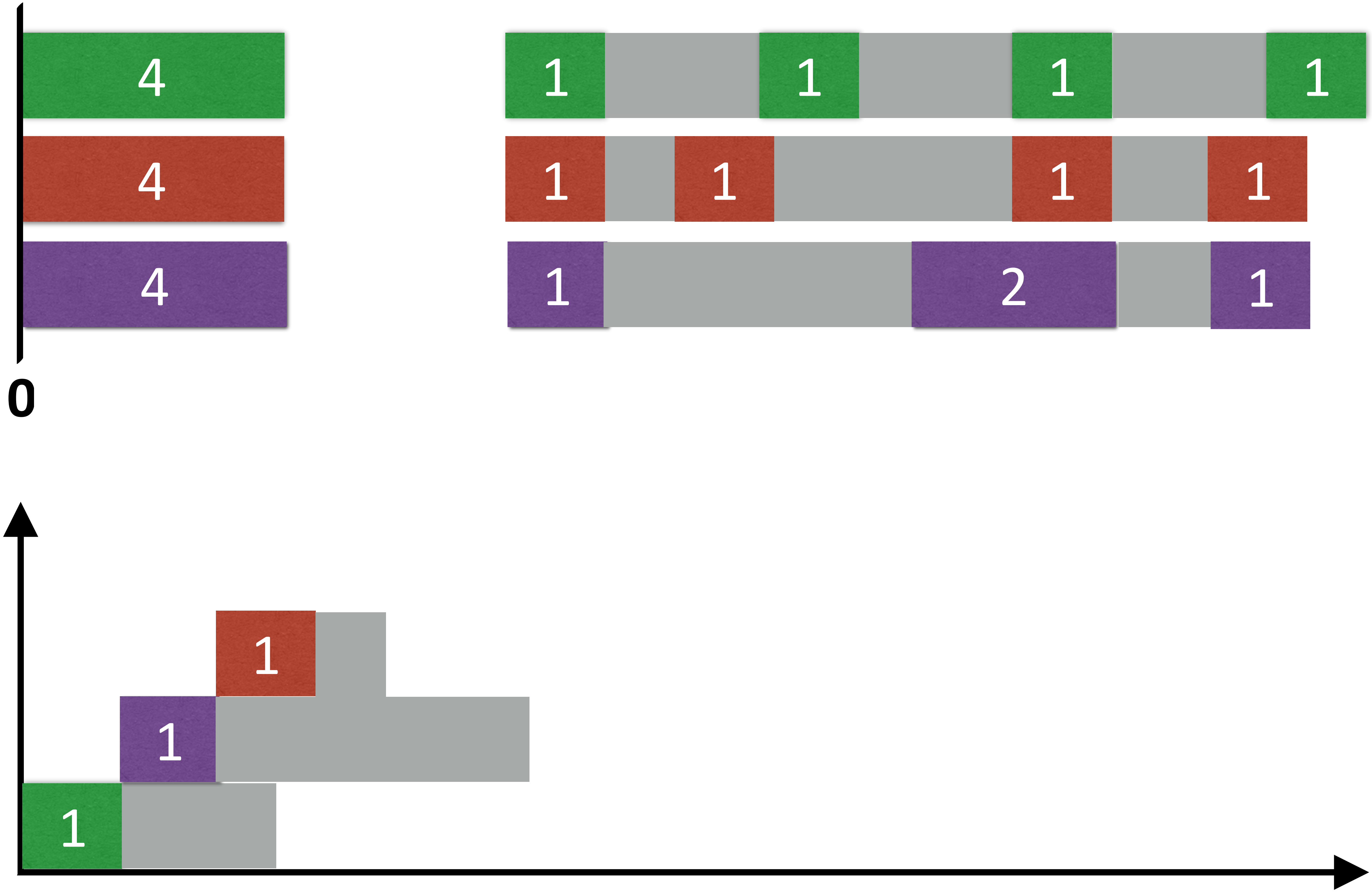


# Improved Resource Utilization — Preemption

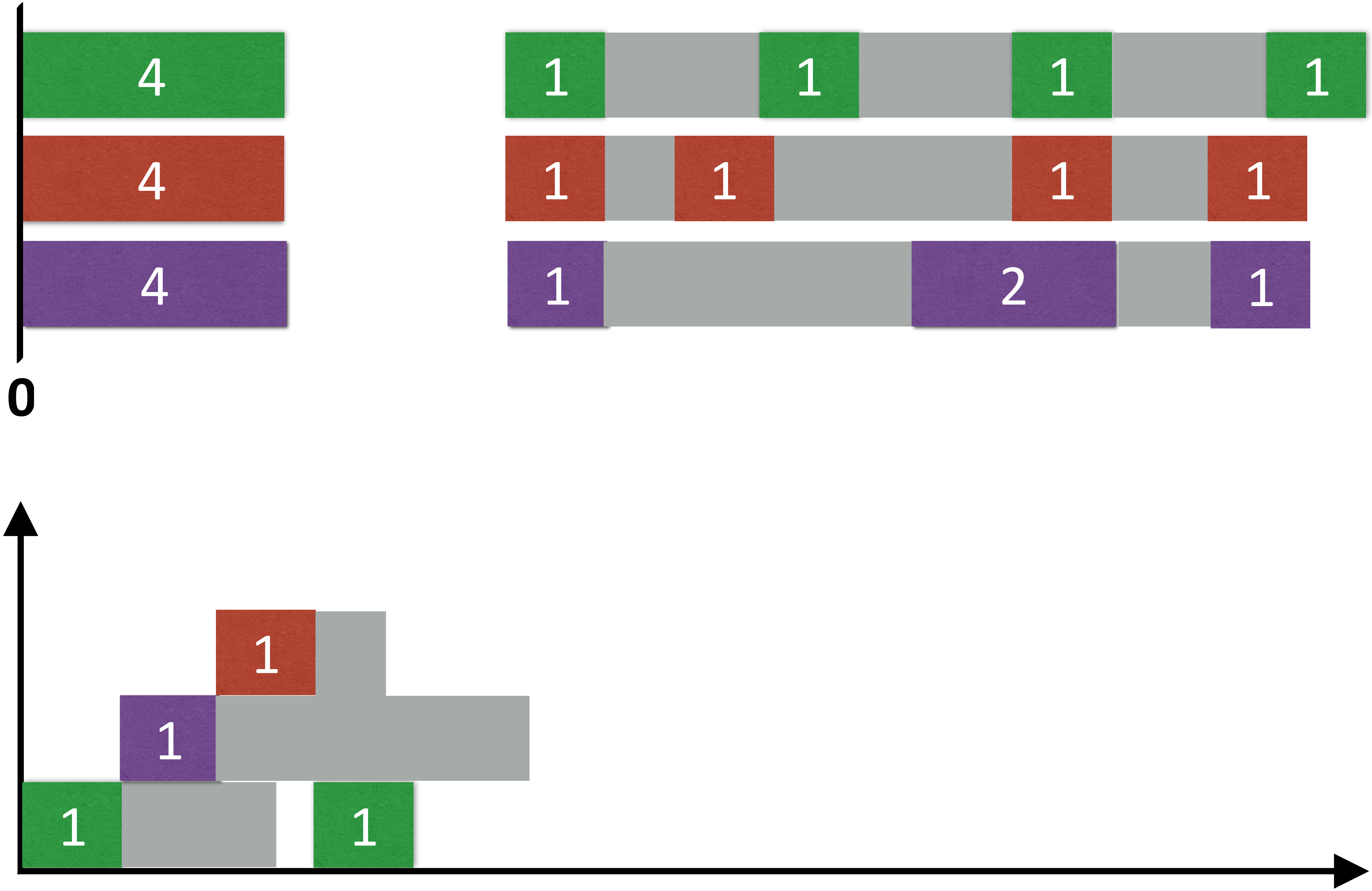



Assumption?

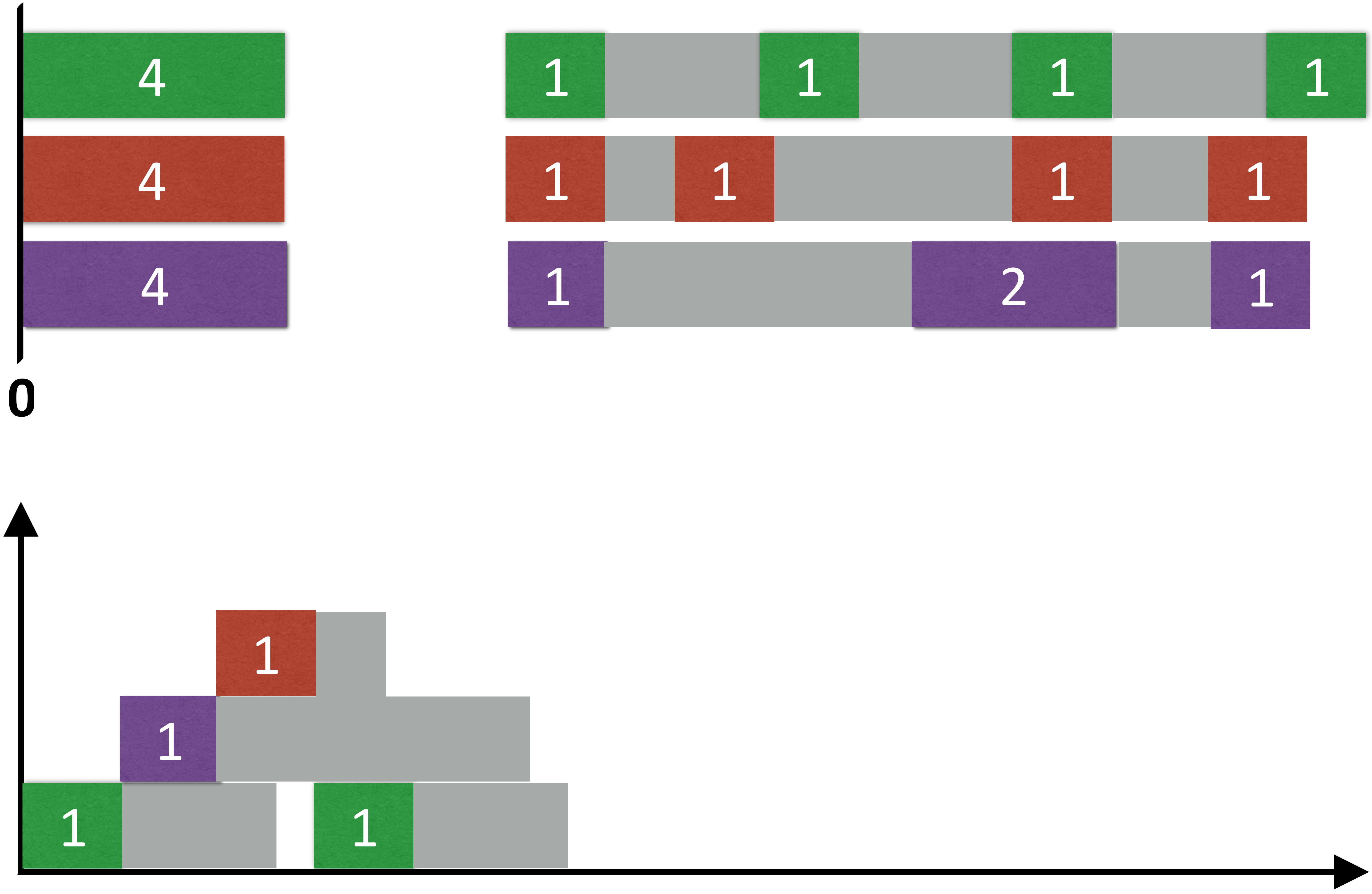
# Improved Resource Utilization — Preemption



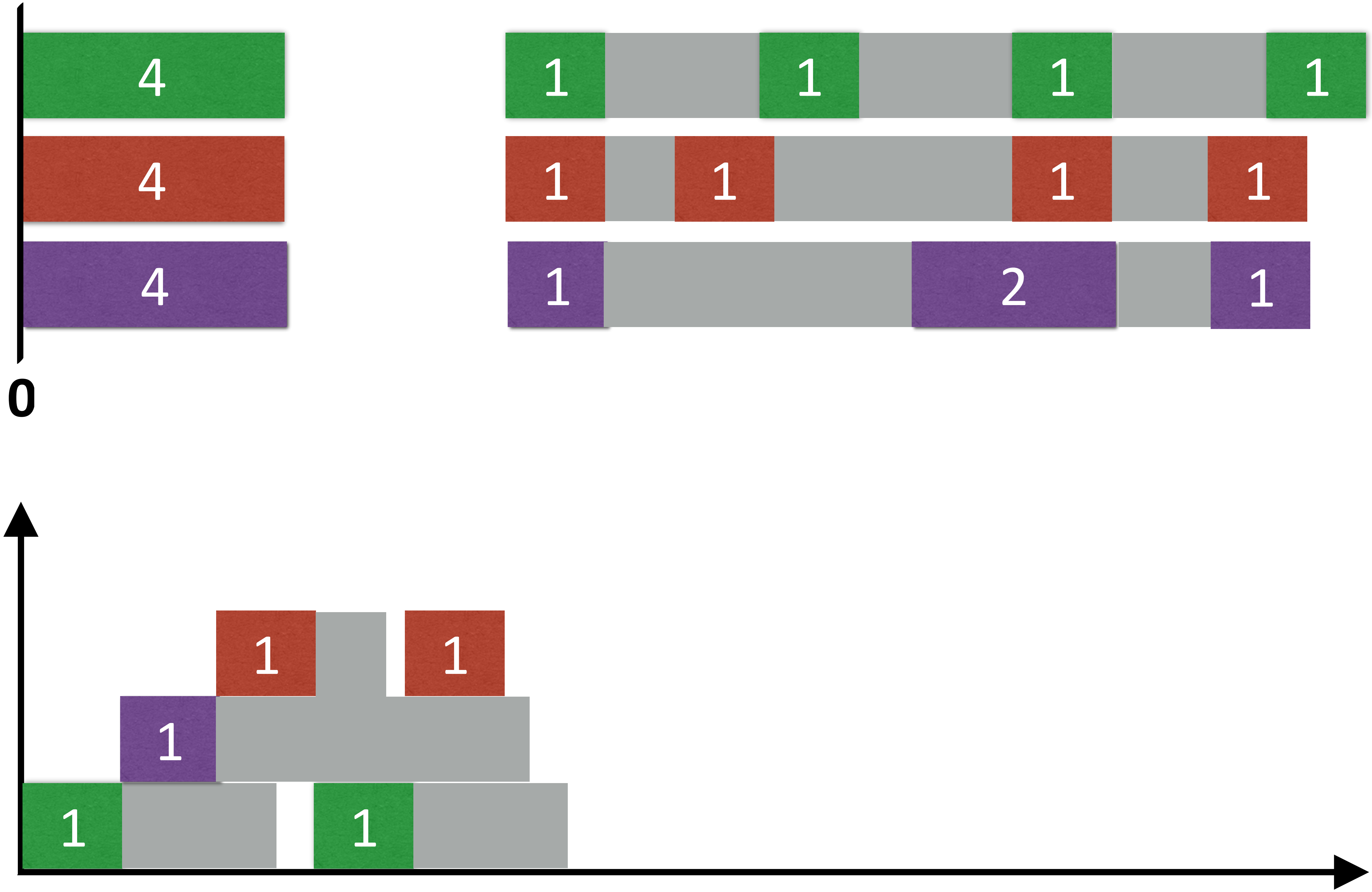
# Improved Resource Utilization — Preemption



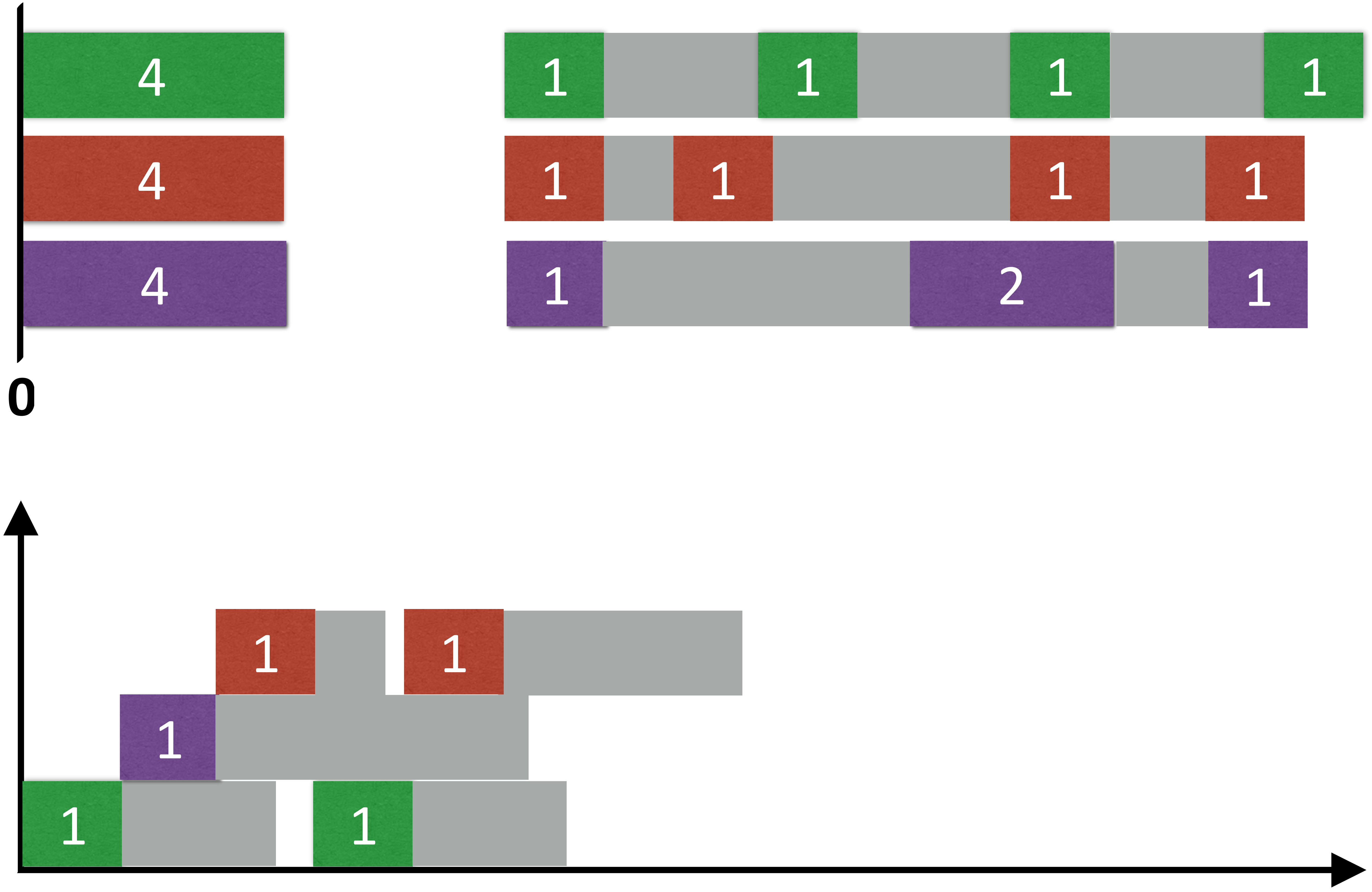
# Improved Resource Utilization — Preemption



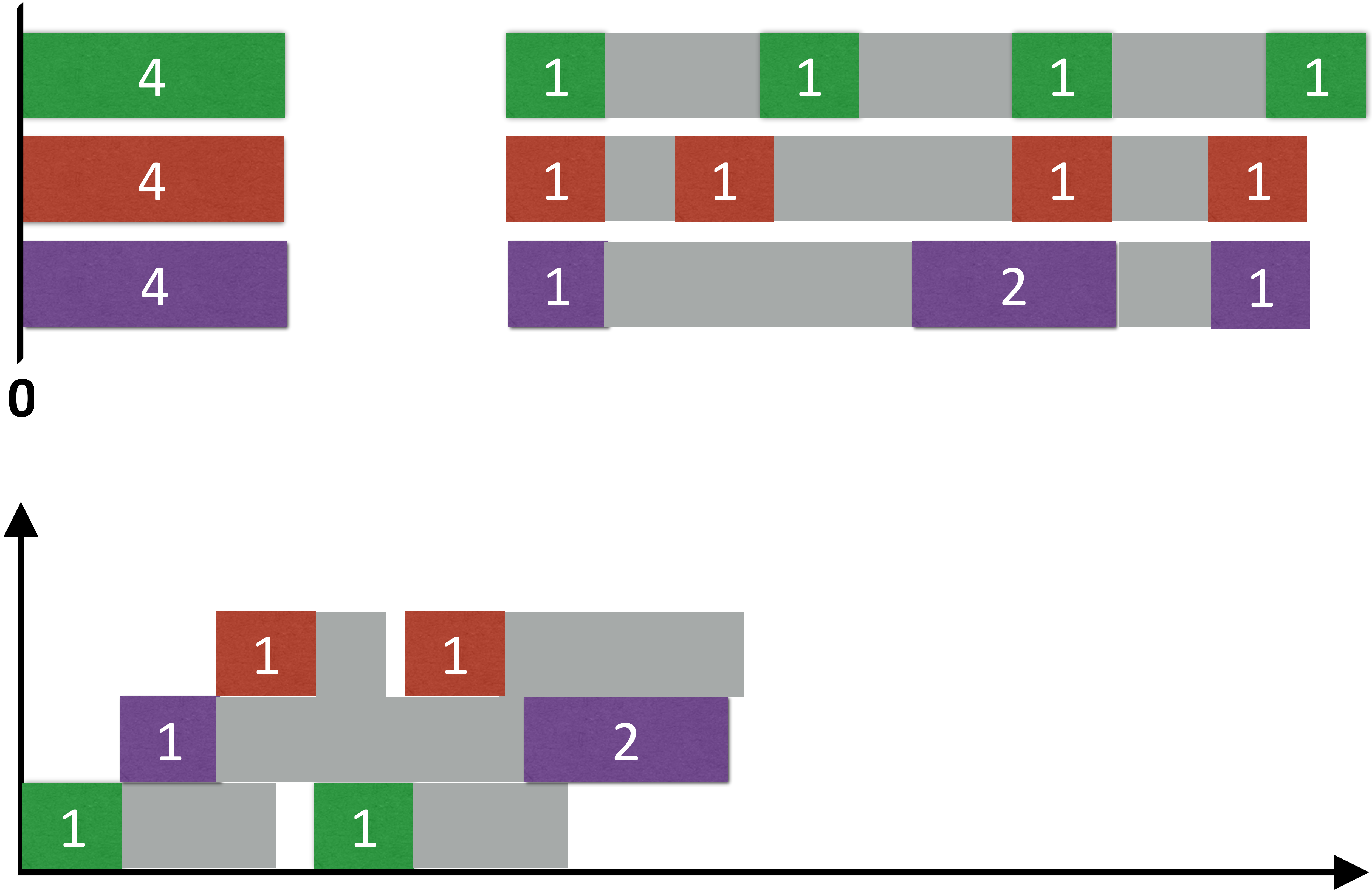
# Improved Resource Utilization — Preemption



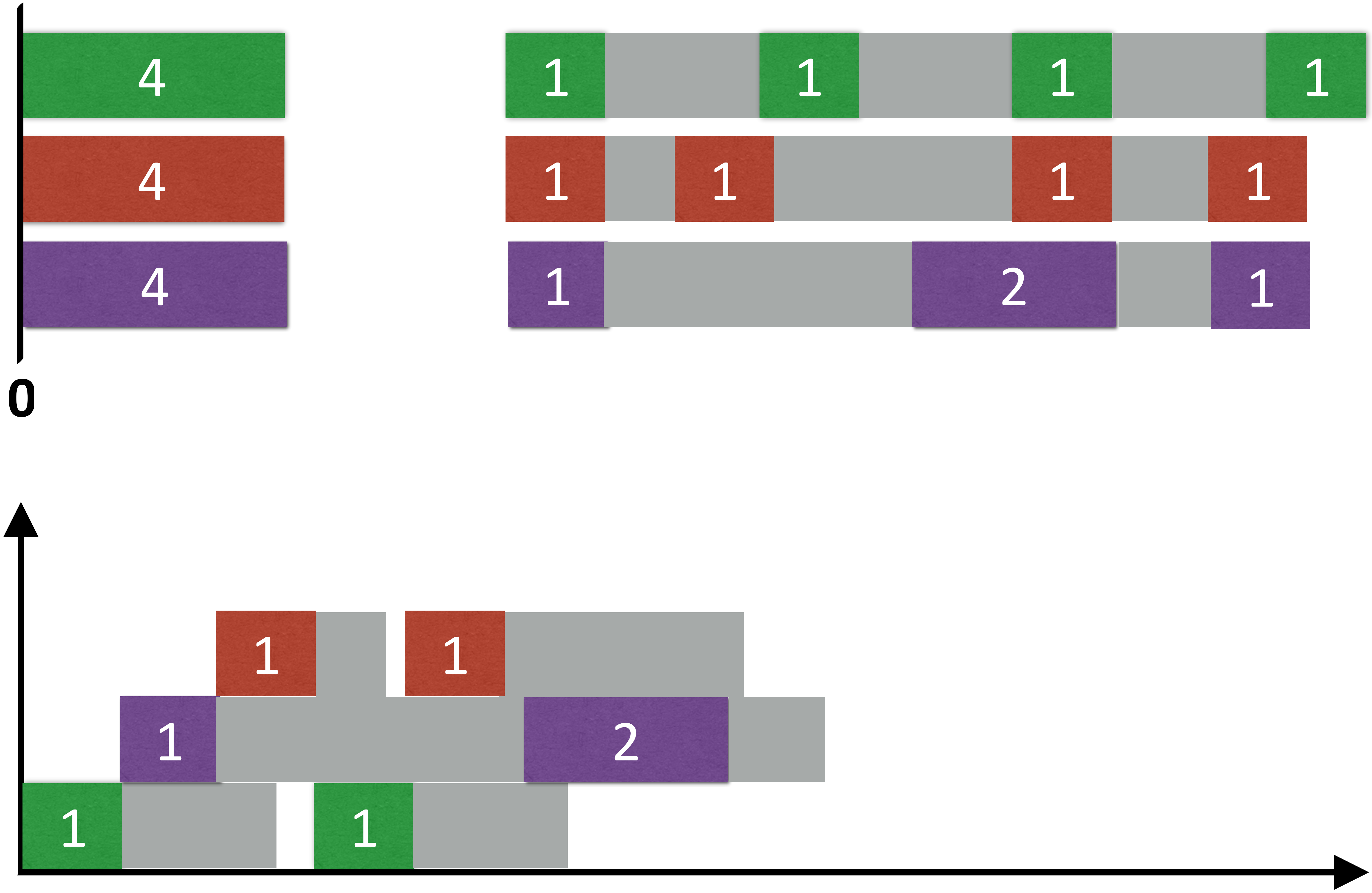
# Improved Resource Utilization — Preemption



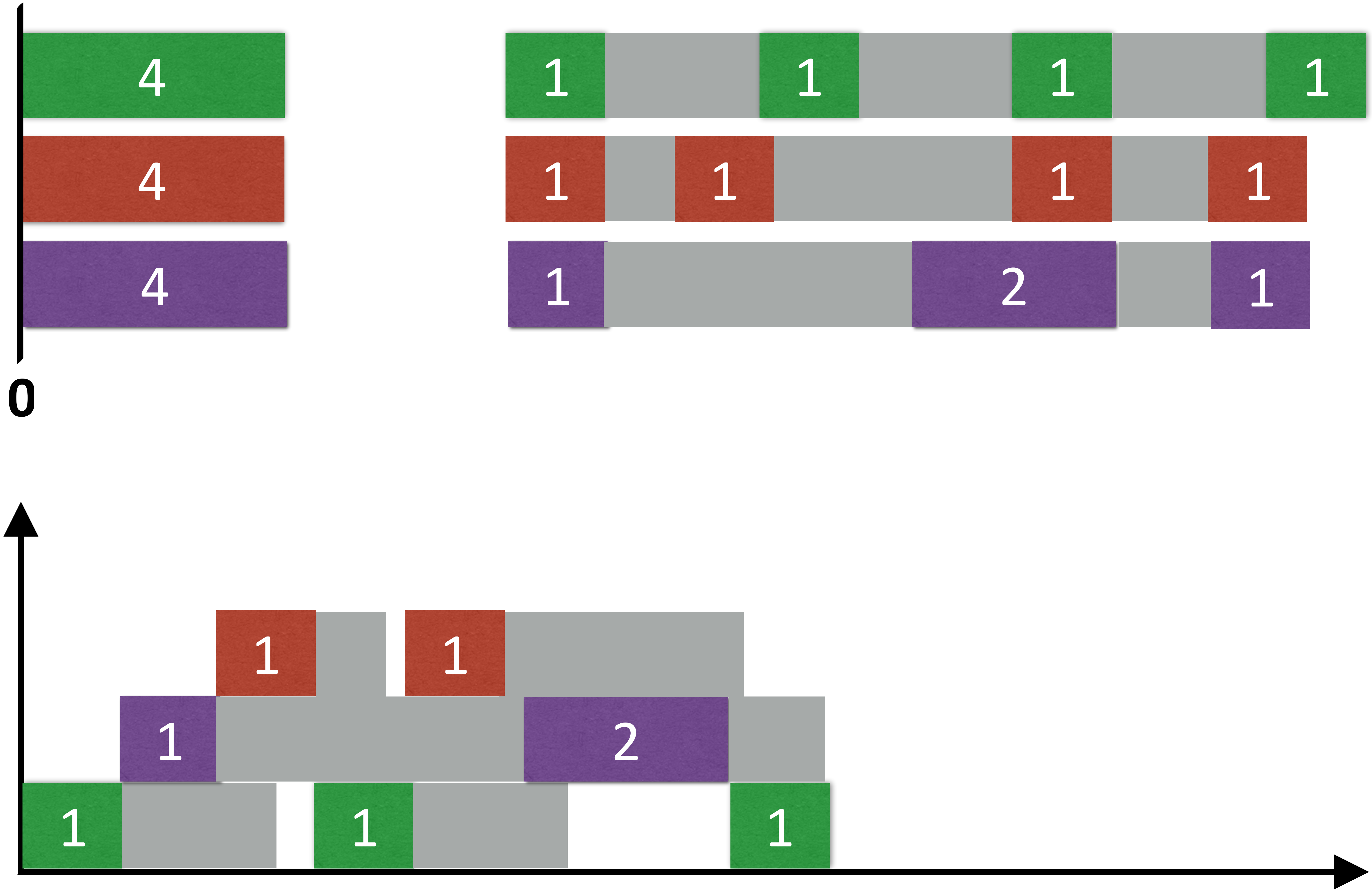
# Improved Resource Utilization — Preemption



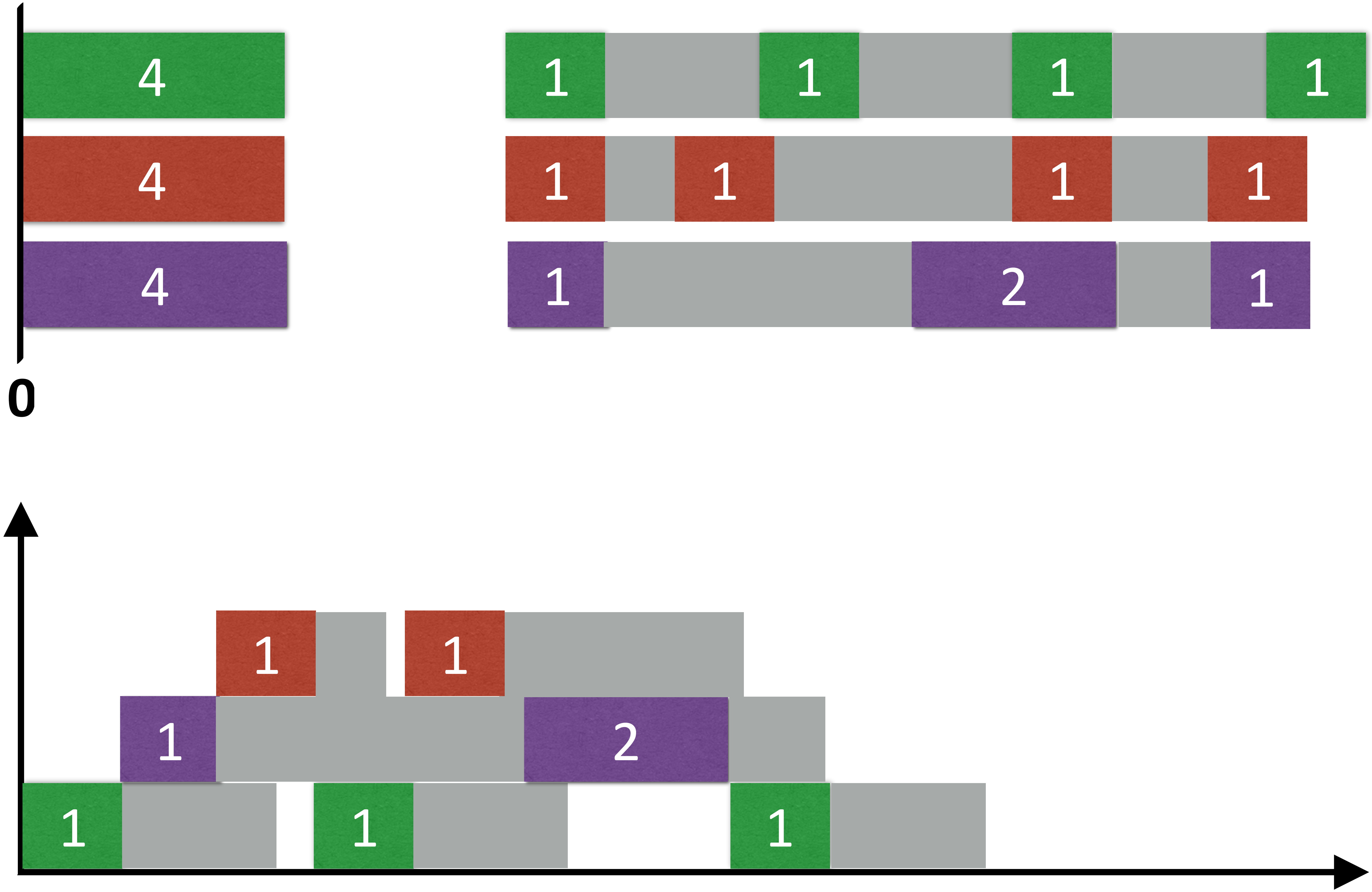
# Improved Resource Utilization — Preemption



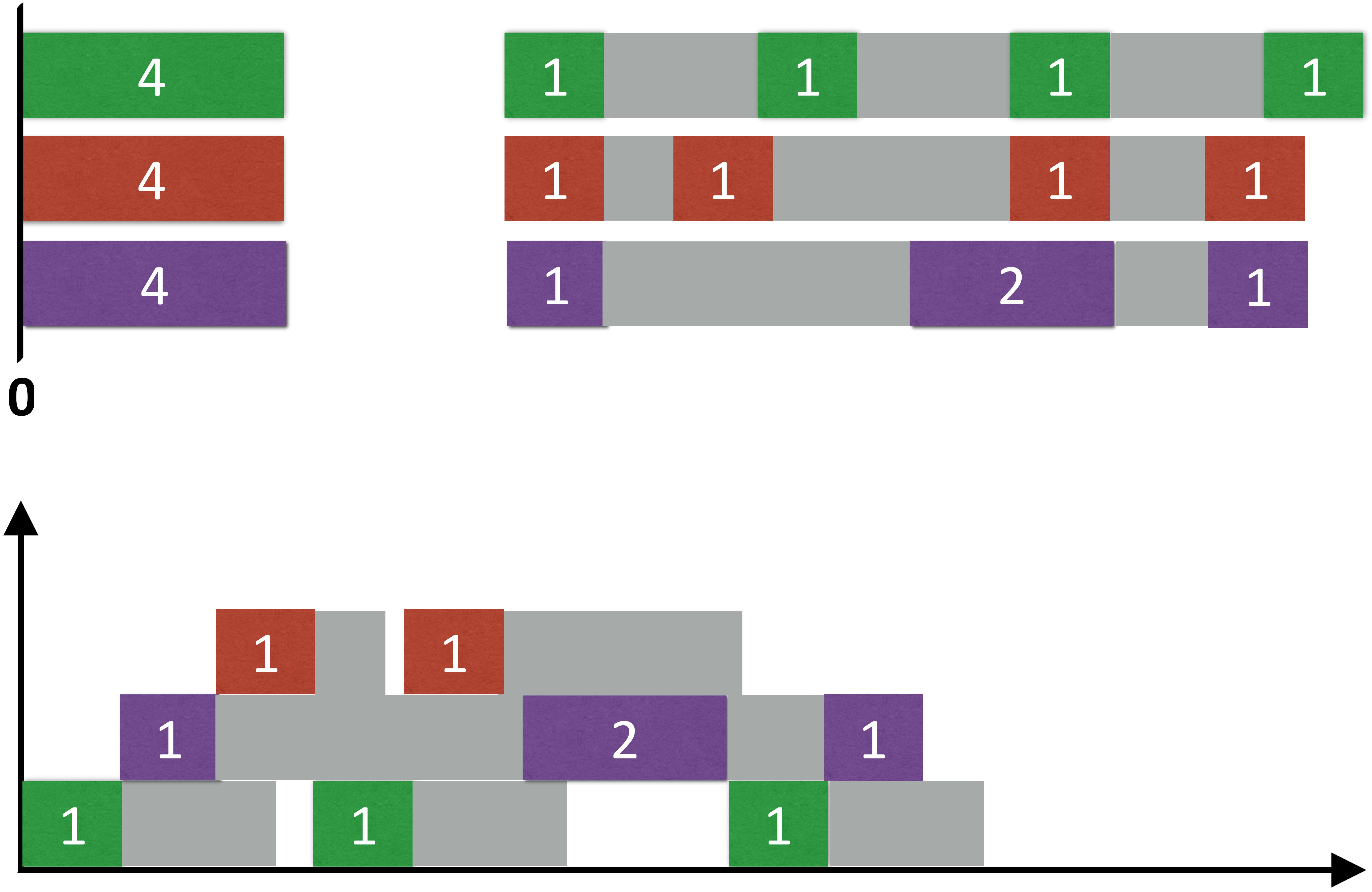
# Improved Resource Utilization — Preemption



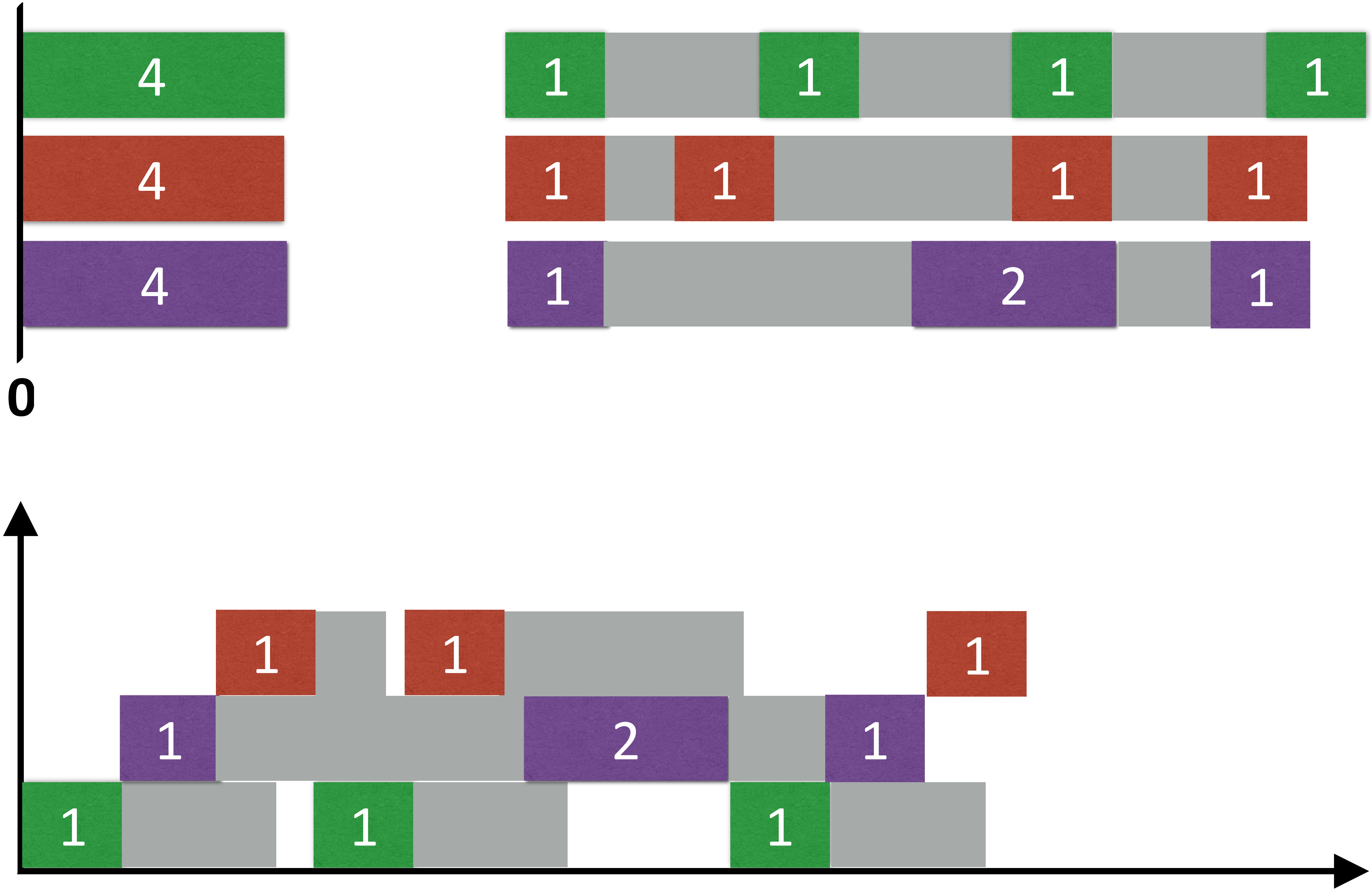
# Improved Resource Utilization — Preemption



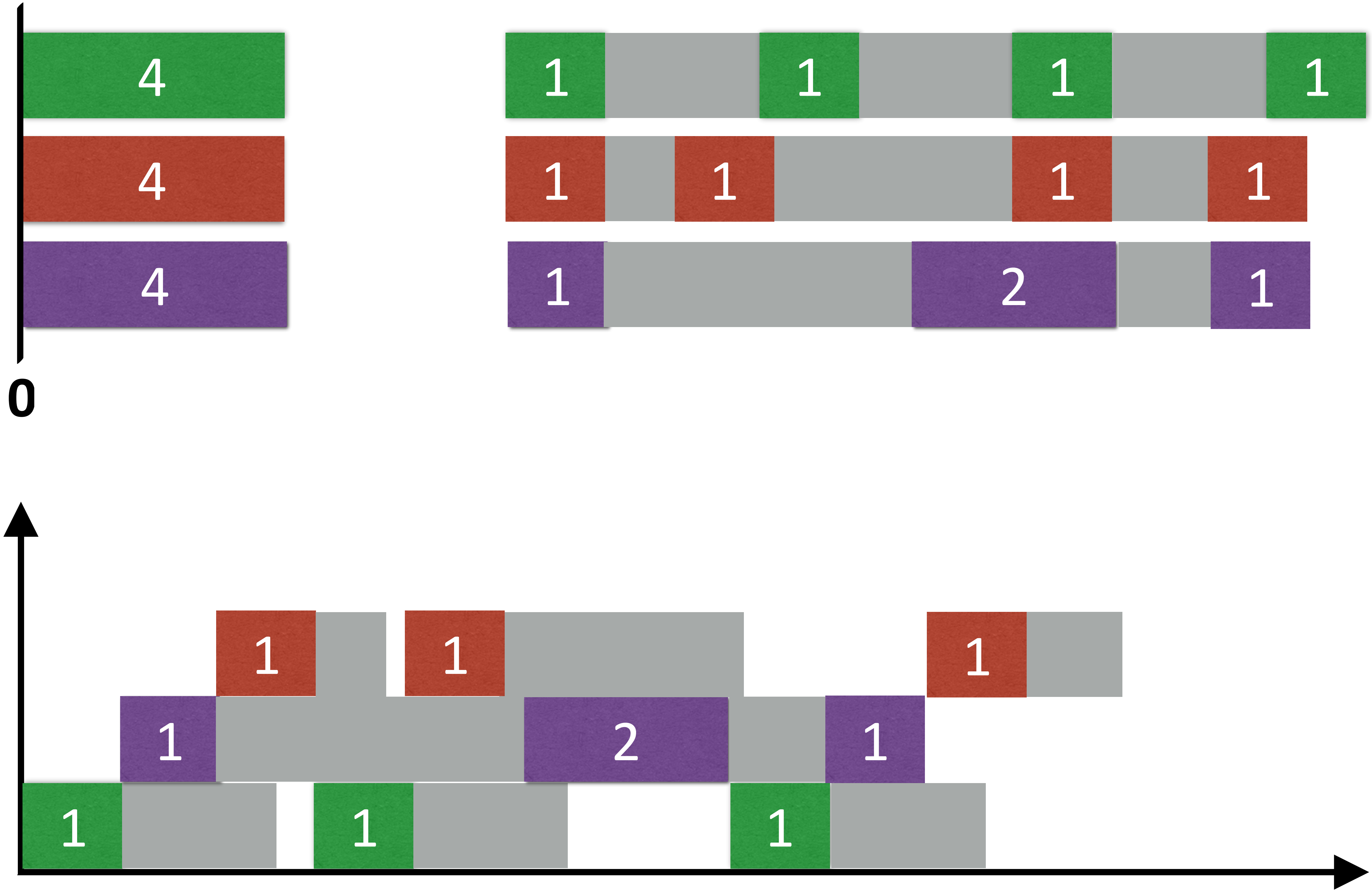
# Improved Resource Utilization — Preemption



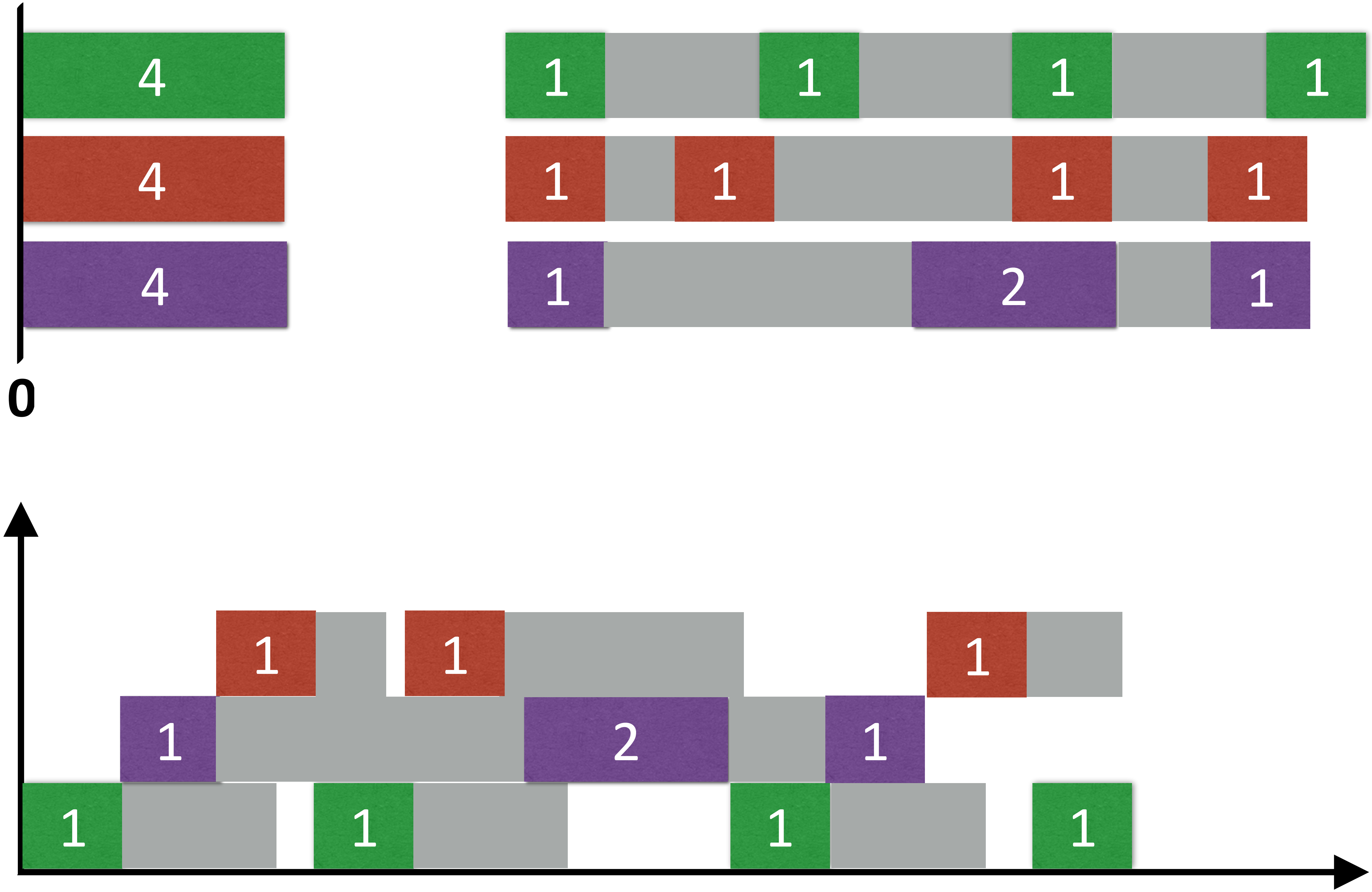
# Improved Resource Utilization — Preemption



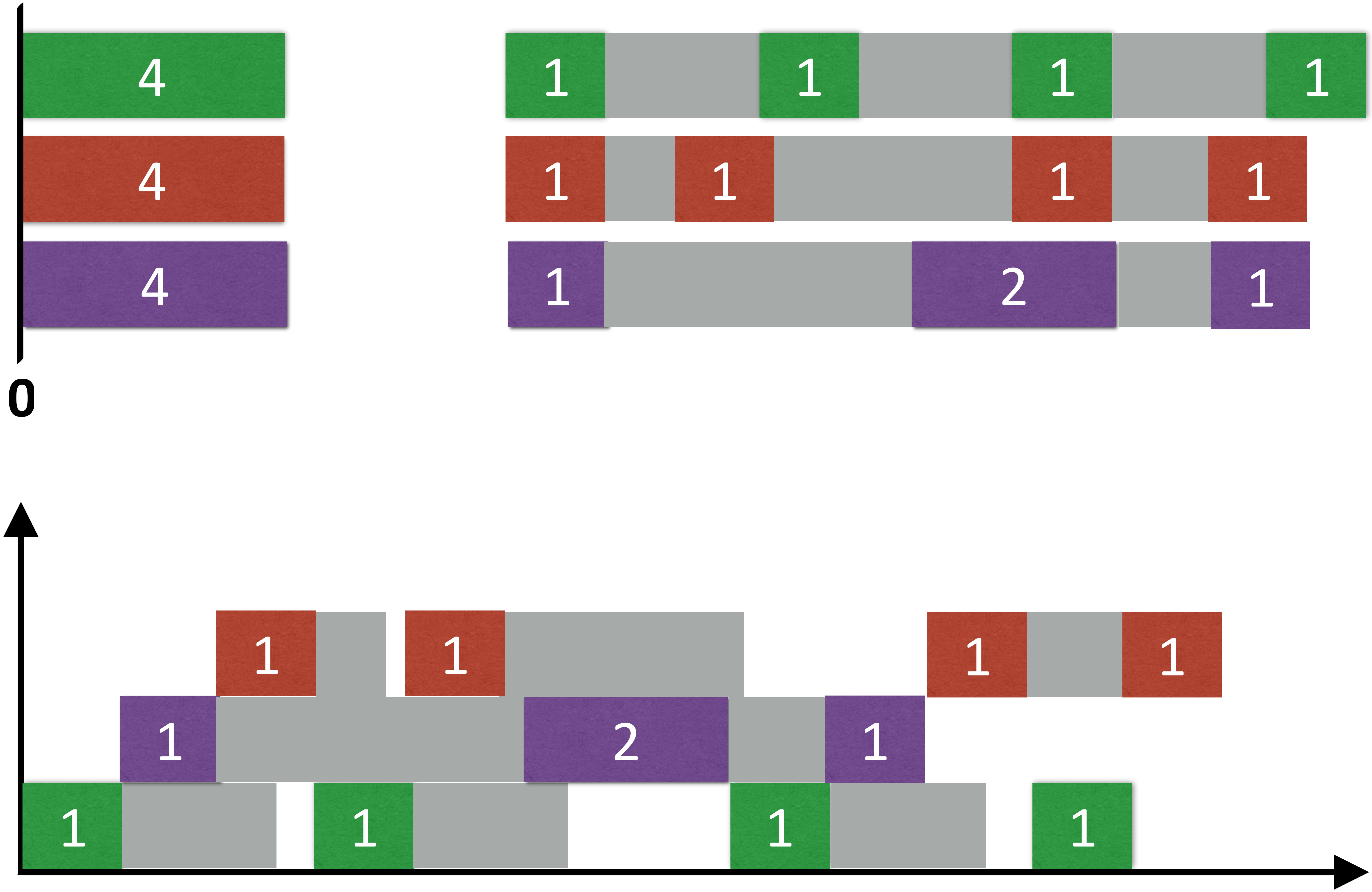
# Improved Resource Utilization — Preemption



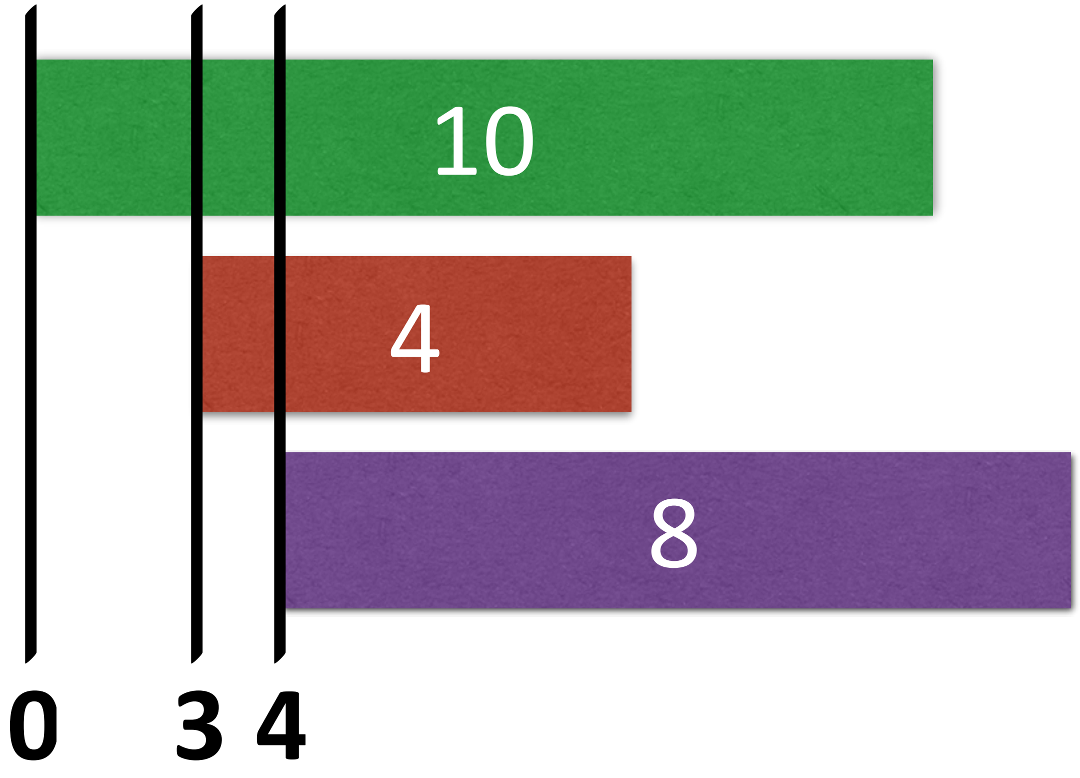
# Improved Resource Utilization — Preemption



# Improved Resource Utilization — Preemption

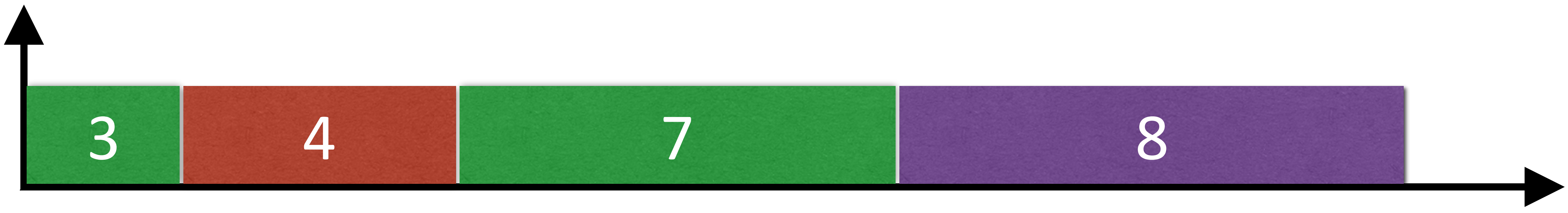


# Scheduling for "Average" Latency — SRTF

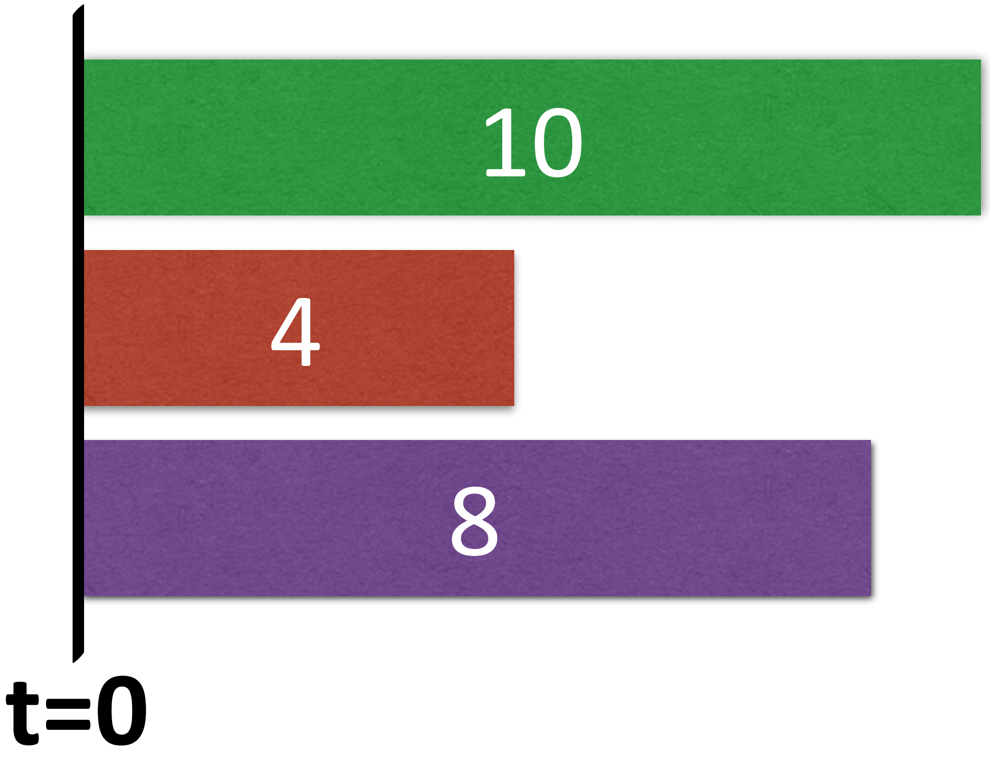


- **Shortest-Remaining Time First**
- **Goods:** Minimizes Avg. ACT
- **Not-so-goods: Starvation**
- **Optimal? Why, or why not?**

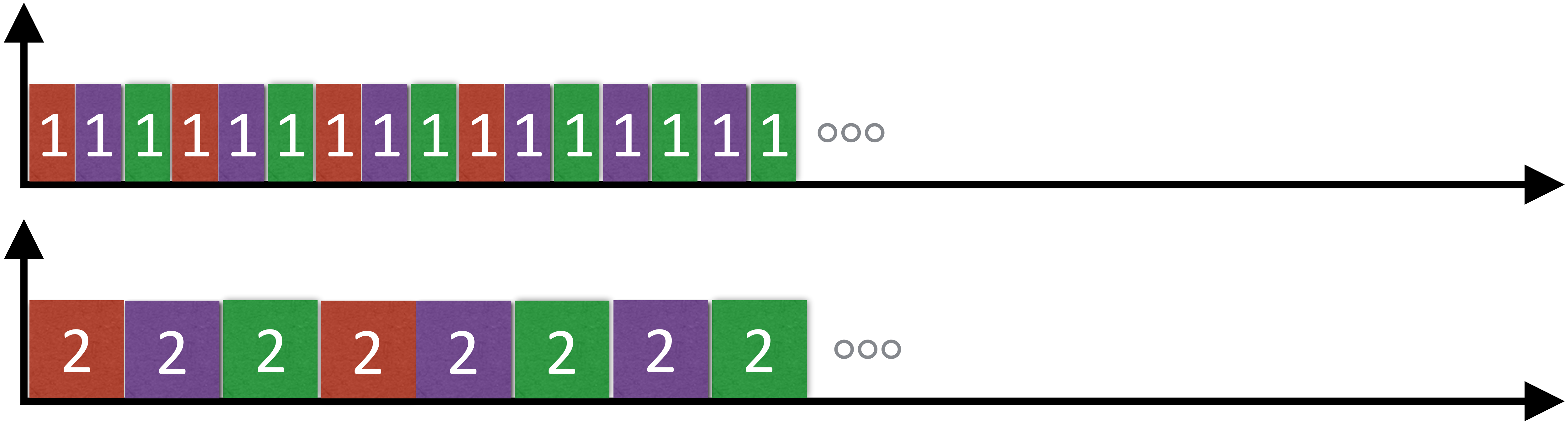
- **Schedule?**



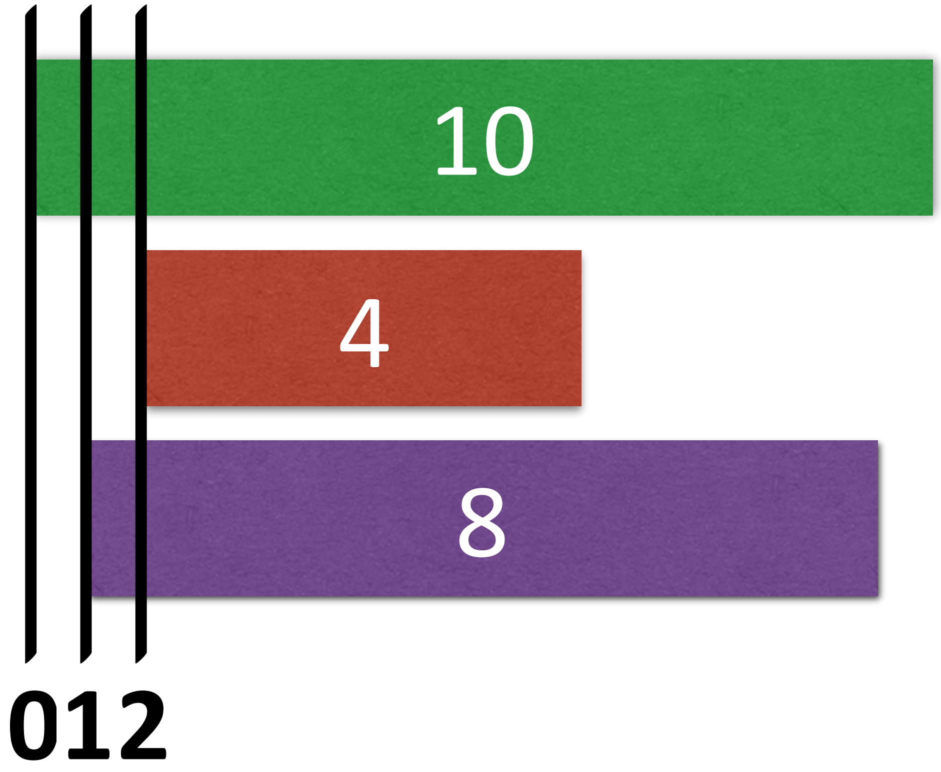
# Scheduling for Fairness — RR



- **Round Robin**



# Scheduling for Deadlines — EDF

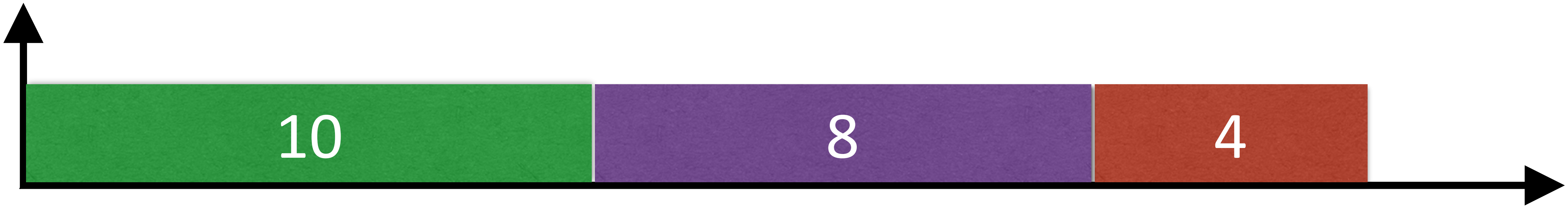


4410 HW (11 hrs)

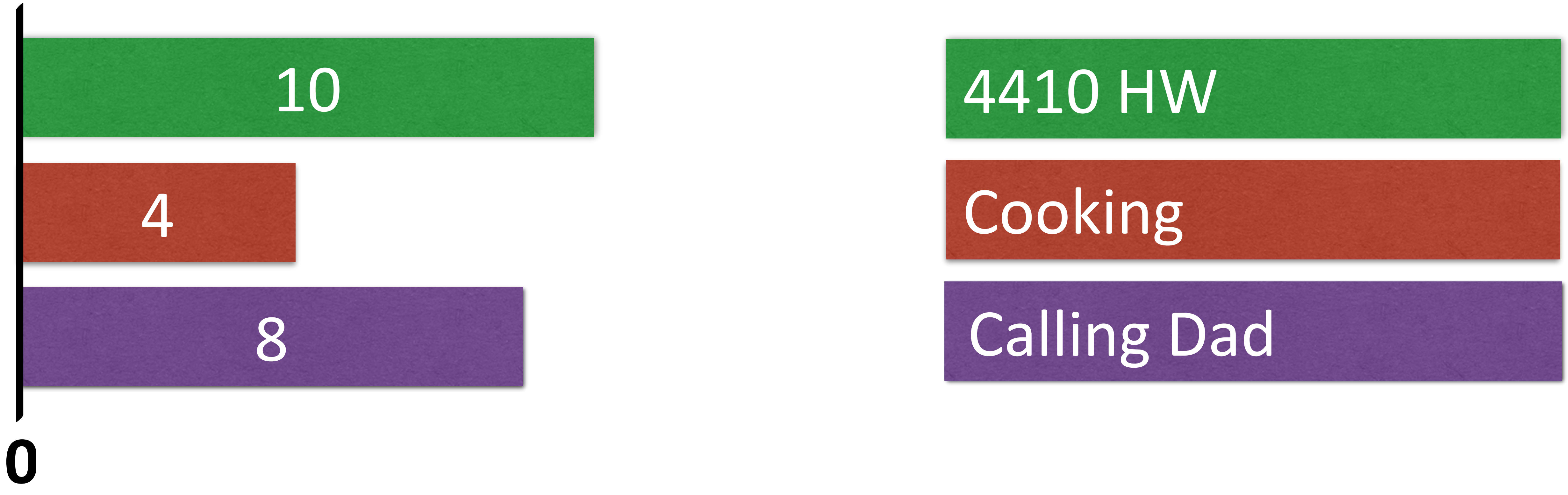
Cooking (23 hrs)

Calling Mom (18 hrs)

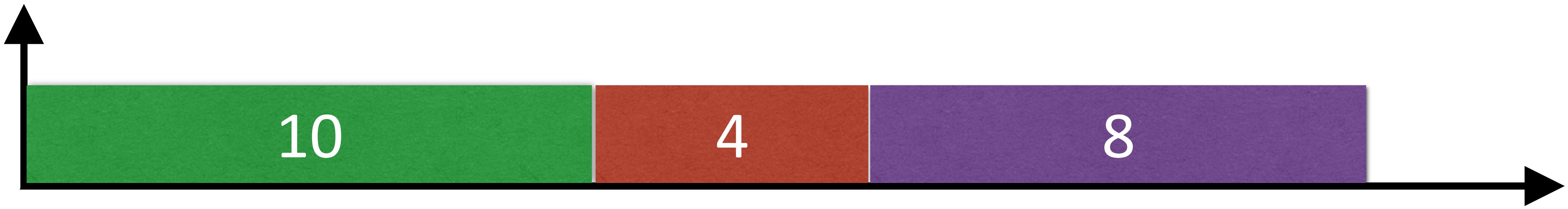
- **Schedule?**



# Scheduling for Happiness



- **Your chosen schedule:**



- **Priority Scheduling**

# “Universal” Scheduling

## Do we need to implement each and every policy?

| | |
|---|---|
| **FIFO** | Arrival time |
| **LIFO** | Current time - Arrival time |
| **SJF** | Job length |
| **SRTF** | Remaining job length |
| **RR** | ? |
| **Priority** | Priority |

# “Mix-and-match” Scheduling

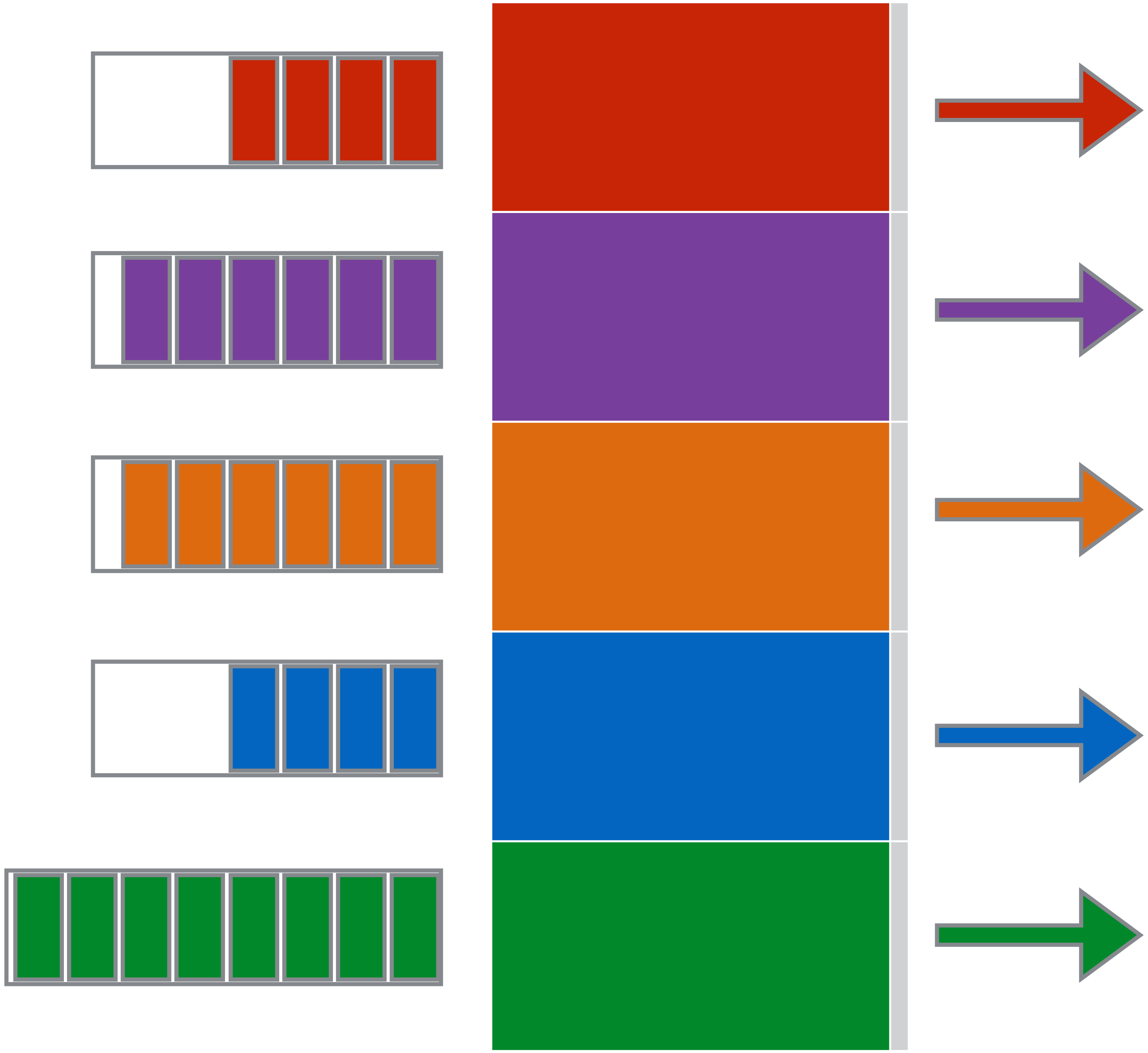
## Different kind of jobs sharing the same OS ….

| | |
|---|---|
| **Interactive** | Facebook, Skype, … |
| **Batch** | Data Analytics |
| **Network bound** | Downloading movies |
| **CPU bound** | Siri, Image processing, … |
| **Low priority** | Life |

# “Mix-and-match” Scheduling

## Different kind of jobs sharing the same OS ….

- Multi-level Queue Scheduling
- Each queue may implement a different policy



# CPU Scheduling — Topics we did not cover

**Many other scheduling problems within OS!**

- **Multi-processor scheduling**
  - How to schedule tasks across multiple processors?
- **Threads vs Processes**
  - How to schedule threads?
- **Jobs with dependencies**
  - Job 2 can run only after Job 1 has finished…
- **….**